

IBM API Connect 10.0.5.x LTS



Tables of Contents

API Connect overview	1
What's new in the latest release (Version 10.0.5.4)	2
What's new in Version 10.0.5.3	3
What's new in Version 10.0.5.2	6
What's new in Version 10.0.5.1	7
What's new in Version 10.0.5.0	9
Language versions offered by API Connect	11
Known limitations	11
API Connect concepts	19
Who does what in API Connect?	19
API Connect user roles	20
Packaging strategy and terminology in API Connect	24
Understanding rate limits for APIs and Plans	28
API Connect components	29
Gateway types	31
API Connect support	32
API Connect glossary	32
Accessibility features for IBM API Connect	35
Legal information	35
Tracking API volume for auditing and compliance	35
Notices	35
Terms and conditions for information centers	37
IBM API Connect Considerations for GDPR Readiness	37
Essential reading	40
Tutorials	41
Installing and maintaining IBM API Connect	41
Kubernetes, OpenShift, and Cloud Pak for Integration	42
Planning your deployment	42
Key concepts: Custom Resources, Operators, and Operands	42
Planning your deployment topology	43
API Connect deployment profiles for OpenShift and Cloud Pak for Integration	44
Component profiles, CPU limits, memory limits, and licensing	45
Management component deployment profile limits	45
Portal component deployment profile limits	51
Analytics component deployment profile limits	53
DataPower gateway deployment profile limits	55
Network requirements for inter-subsystem communication	55
Enable JWT security instead of mTLS	56
In-cluster service communication between subsystems	58
Kubernetes ingress controller prerequisites	58
Persistent Volume (PV) requirements	59
Load balancer configuration in a Kubernetes deployment	60
Firewall requirements on Kubernetes	62
Planning your analytics deployment	65
Planning the analytics profile, storage class, and storage type	65
Planning your topology	66
Analytics pods and PVCs	66
Planning to offload data to a third-party system	66
Planning to disable internal storage	67
Planning to modify your incoming analytics data	67
Estimating internal storage space	67
Two data center deployment strategy on Kubernetes and OpenShift	69
Key Concepts: Cert-manager, Issuers, and Secrets	70
Certificates in a Kubernetes environment	72
Custom certificates on Kubernetes	76
Custom certificates reference	76
Generating custom certificates	78
Generating custom certificates using cert-manager in a two data center deployment	80
Configuring custom certificates before installation	82
Converting to custom front-end/ingress certificates after deployment	84
Webhooks deployed and managed by the API Connect Operator	84
Installing API Connect	85
Deployment requirements	86
IBM API Connect Version 10 software product compatibility requirements	89
API Connect licenses	90
Deployment procedures	91
Deploying on Kubernetes	91
Obtaining product files	91
Signature verification by using PGP	93
Deploying operators and cert-manager	93

Deploying operators in a single-namespace API Connect cluster	94
Deploying operators in a multi-namespace API Connect cluster	96
Installing cert-manager and certificates in a two data center deployment	99
Installing the API Connect subsystems	100
Installing the Management subsystem cluster	101
Installing the DataPower Gateway subsystem	103
Installing the Developer Portal subsystem	106
Defining multiple portal endpoints for a Kubernetes environment	109
Installing the Analytics subsystem	110
Creating the analytics CR	110
Topology settings and additional analytics features	112
Analytics persistent queue	112
Offloading data to a third-party system	113
Sample output plugins for offloading data	114
Providing a custom certificate for Analytics offload	115
Disabling internal storage	116
Modifying incoming analytics data	117
Sample filters for modifying analytics data	118
Reference: Fields in the Analytics CR	119
Installing analytics	121
Installing a two data center deployment on Kubernetes	122
Converting a single data center to a two data center deployment on Kubernetes	125
Deploying on OpenShift and Cloud Pak for Integration	129
Preparing for installation	130
Operator, operand, and CASE versions	130
API Connect configuration settings	130
Basic configuration settings	131
Management subsystem settings	132
Gateway subsystem settings	134
Developer Portal subsystem settings	136
Analytics subsystem settings	137
Analytics subsystem CR configured settings	138
Dedicated storage instead of shared storage	138
Analytics persistent queue	139
Offloading data to a third-party system	139
Sample output plugins for offloading data	140
Providing a custom certificate for Analytics offload	141
Disabling internal storage	143
Modifying incoming analytics data	143
Sample filters for modifying analytics	144
Reference: Fields in the Analytics CR	146
Additional Kubernetes settings	147
Configuring timeouts for management endpoints	148
Installation procedures with IBM Cloud Pak for Integration	149
Installing with the top-level CR on OpenShift	149
Connected installation	149
Installing operators	150
Installing API Connect	152
Air-gapped installation	153
Installing with a bastion host	154
Installing with a portable computer or storage device	159
Enable additional features post-install	164
Enable management CA verification on REST API calls	164
Enable JWT security and disable mTLS between subsystems	165
Installing with subsystem CRs in a shared namespace on OpenShift	166
Installing operators	166
Setting up a certificate issuer	168
Installing the Management subsystem in a shared namespace	171
Installing the Gateway subsystem in a shared namespace	174
Installing the Portal subsystem in a shared namespace	177
Installing the Analytics subsystem in a shared namespace	180
Installing with subsystem CRs in different namespaces or environments on OpenShift	182
Installing operators	183
Installing the Management subsystem	184
Extracting the Management ingress-ca certificates	188
Installing the Gateway subsystem	189
Installing the Portal subsystem	193
Installing the Analytics subsystem	197
Installing a two data center deployment with Cloud Pak for Integration	200
Planning and initial preparation	200
Preparing your active data center	201
Preparing your warm-standby data center	204

Installing API Connect on the active data center	206
Installing API Connect on the warm-standby data center	208
Installing a two data center deployment on OpenShift	210
Planning and initial preparation	211
Preparing your active data center	211
Preparing your warm-standby data center	213
Installing API Connect on the active data center	218
Installing API Connect on the warm-standby data center	219
Troubleshooting installation and upgrade on OpenShift	221
Upgrading API Connect	225
Upgrading on Kubernetes	226
Upgrade considerations on native Kubernetes	226
Additional considerations for upgrading analytics from a version prior to 10.0.5.0	229
Exporting analytics data from the UI	229
Exporting aggregated analytics data from a visualization	230
Exporting analytics data by using the REST API	231
Offloading analytics data	232
Remove template overrides from your analytics CR	232
Upgrading subsystems on native Kubernetes	233
Upgrading Management subsystem on native Kubernetes	237
Upgrading Portal subsystem on native Kubernetes	238
Upgrading Analytics subsystem on native Kubernetes	239
Upgrading Gateway subsystem on native Kubernetes	240
Enabling gateway peering and verifying cluster status	241
Enabling Analytics after upgrading	242
Optional post-upgrade steps for upgrade to 10.0.5.3 (or later) from earlier 10.0.5 release	243
Upgrading from 2018 on native Kubernetes	245
Preparing the 2018 deployment for upgrade	245
Obtaining files for upgrade from 2018	246
Deploying operators in a single-namespace API Connect cluster	248
Deploying operators in a multi-namespace API Connect cluster	249
Upgrading the Management subsystem from 2018 to v10	252
Rolling back a management upgrade to v2018	255
Cleaning up a failed management upgrade	256
Stopping the v2018 to v10 management upgrade	257
Troubleshooting the management subsystem upgrade from v2018 to v10	257
Upgrading the Developer Portal subsystem from 2018 to v10	259
Rolling back Developer Portal to v2018	261
Troubleshooting Portal subsystem upgrade from v2018 to v10	262
Upgrading the Analytics subsystem from v2018 to v10	262
Upgrading the gateway subsystem from 2018 to v10	263
Rollback of gateway subsystem to v2018	265
Post-upgrade steps for upgrading to v10	265
Disaster recovery of a 2018 deployment	267
Troubleshooting upgrades on Kubernetes	267
Migrating API test data	269
Backing up API test data	270
Restoring API test data	271
Uninstalling Percona Server for MongoDB	272
Upgrading on OpenShift and Cloud Pak for Integration	273
Upgrading API Connect in Cloud Pak for Integration	273
Upgrade considerations on OpenShift	274
Additional considerations for upgrading to analytics from a version prior to 10.0.5.0	276
Exporting analytics data from the UI	277
Exporting aggregated analytics data from a visualization	277
Exporting analytics data by using the REST API	279
Offloading analytics data	280
Remove template overrides from your analytics CR	280
Preparing to upgrade on OpenShift	281
Upgrading on OpenShift in an online environment	282
Air-gapped upgrade	285
Upgrading with a bastion host	286
Upgrading with a portable computer or storage device	292
Upgrading from 2018 on OpenShift	299
OpenShift: Preparing the 2018 deployment for upgrade	299
OpenShift: Upgrading from 2018 to v10	300
OpenShift: Post-upgrade steps	303
OpenShift: Recovering the V2018 deployment	305
Optional post-upgrade steps for upgrade to 10.0.5.3 from earlier 10.0.5 release	306
Enabling Analytics after upgrading	309
Troubleshooting installation and upgrade on OpenShift	309
Upgrading a two data center deployment on Kubernetes and OpenShift	313

Upgrading a two data center deployment on Cloud Pak for Integration	316
Maintaining API Connect	317
Advanced configuration	318
Installing License Service	318
Installing the Automated API behavior testing application	319
Installing the toolkit	320
Changing deployment profiles on Kubernetes	322
Changing deployment profiles on OpenShift	323
Replica override for microservices	324
Changing the deployment profile on Cloud Pak for Integration	325
Enabling Developer Portal feature flags on Kubernetes	325
Enabling Developer Portal feature flags on OpenShift and Cloud Pak for Integration	326
Enabling UI feature flags on Kubernetes	327
Enabling UI feature flags on OpenShift and Cloud Pak for Integration	328
Advanced configuration for the management subsystem	329
Configuring maximum size of client requests to the Management subsystem	329
Setting rate limits for public APIs on the management service for a Kubernetes environment	330
Configuring monetization on Kubernetes	331
Overriding resources for Postgres components on Kubernetes	332
Overriding resources for Postgres components on OpenShift	333
Advanced configuration for the analytics subsystem	334
Changing storage type between shared and dedicated	335
Horizontal scaling of analytics data storage pod replicas	335
Advanced configuration for the gateway subsystem	338
Customizing a DataPower deployment	339
Overriding the default DataPower image, version, and license	339
Enabling autoscaling of gateway pods	340
Dynamically re-registering and reconfiguring a Gateway service in a Kubernetes deployment	340
Backing up and restoring	341
Backups on Kubernetes	342
Backing up and restoring the management subsystem	342
Configuring backup settings for fresh install of the Management subsystem	342
Configuring S3 backup settings for fresh install of the Management subsystem	342
Configuring SFTP backup settings for fresh install of the Management subsystem	345
Configuring local backup settings for fresh install of the Management subsystem	346
Reconfiguring or adding backup settings after installation of the management subsystem	346
Generating a manual backup of the management subsystem	347
Restoring the management subsystem (v10.0.1.1 or later)	349
Troubleshooting management subsystem backups	350
Troubleshooting resiliency issues	350
Troubleshooting stanza-create job for S3 backup configuration	350
Restore failure with bootstrap pod error	351
Backing up and restoring the Developer Portal in a Kubernetes environment	352
Overview of the Developer Portal backup resources	359
Backing up and restoring the analytics database	359
Configuring backup settings for Analytics	359
Running a backup of the Analytics database	361
Restoring the Analytics database	362
Troubleshooting Analytics backup and restore	363
Backups on OpenShift	364
Backing up and restoring the management subsystem on OpenShift and Cloud Pak for Integration	364
Configuring backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration	365
Configuring S3 backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration	365
Configuring SFTP backup settings for fresh install of the Management subsystem on OpenShift and Cloud Pak for Integration	368
Configuring local backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration	369
Reconfiguring or adding backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration	370
Generating a manual backup of the management subsystem on OpenShift and Cloud Pak for Integration	371
Restoring the management subsystem on OpenShift and Cloud Pak for Integration	372
Troubleshooting management subsystem backups on OpenShift and Cloud Pak for Integration	374
Troubleshooting resiliency issues on OpenShift and Cloud Pak for Integration	374
Troubleshooting the stanza-create job for S3 backup configuration on OpenShift and Cloud Pak for Integration	374
Backing up and restoring Developer Portal on OpenShift and Cloud Pak for Integration	375
Configuring backups for Developer Portal on OpenShift and Cloud Pak for Integration	376
Overview of the Developer Portal backup resources	359
Backing up Developer Portal on OpenShift and Cloud Pak for Integration	379
Restoring Developer Portal on OpenShift and Cloud Pak for Integration	381
Troubleshooting Developer Portal backup and restore on OpenShift and Cloud Pak for Integration	383
Backing up and restoring the Analytics database on OpenShift and Cloud Pak for Integration	385
Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration	385
Running a backup of the Analytics database on OpenShift and Cloud Pak for Integration	387
Restoring the Analytics database on OpenShift and Cloud Pak for Integration	388
Disaster recovery	389

Disaster recovery on Kubernetes	390
Preparing for a disaster	390
Preparing the management subsystem for disaster recovery	390
Preparing the Developer Portal for disaster recovery	392
Preparing the Analytics subsystem for disaster recovery	393
Recovering from a disaster	393
Recovering the management subsystem on Kubernetes	393
Recovering the management subsystem from S3 backups	394
Recovering the management subsystem from SFTP backups	396
Recovering the Developer Portal after a disaster	398
Recovering the Analytics subsystem after a disaster	399
Disaster recovery on OpenShift and Cloud Pak for Integration	400
Preparing for disaster recovery on OpenShift	400
Preparing the Management subsystem for disaster recovery on OpenShift	401
Preparing the Portal subsystem for disaster recovery on OpenShift	402
Preparing the Analytics subsystem for disaster recovery on OpenShift	403
Recovering from a disaster on OpenShift	403
Configuring the OIDC credentials on OpenShift	407
Preparing for disaster recovery on Cloud Pak for Integration	408
Preparing the Management subsystem for disaster recovery on Cloud Pak for Integration	408
Preparing the Portal subsystem for disaster recovery on Cloud Pak for Integration	410
Preparing the Analytics subsystem for disaster recovery on Cloud Pak for Integration	411
Recovering from a disaster on Cloud Pak for Integration	411
Configuring the OIDC credentials on Cloud Pak for Integration	416
Disabling and re-enabling the Analytics subsystem on Kubernetes	417
Disabling and re-enabling the Analytics subsystem on OpenShift and Cloud Pak for Integration	418
Maintaining a two data center deployment	420
Key concepts of 2DCDR and failure scenarios	420
How to failover API Connect from the active to the warm-standby data center	421
Recovering from a failover of a two data center deployment	423
Backup and restore requirements for a two data center deployment	424
Removing a two data center deployment	425
Verifying replication between data centers	425
Troubleshooting a two data center deployment	427
Monitoring and health checks	428
Monitoring on Kubernetes	429
Monitoring subsystems on Kubernetes	429
Monitoring Postgres disk usage	429
Monitoring platform health using Kubernetes events	431
Monitoring on OpenShift	432
Monitoring the API Connect cluster on OpenShift	432
Monitoring Postgres disk usage on OpenShift	434
Monitoring platform health on Red Hat OpenShift using Kubernetes events	436
Monitoring on Cloud Pak for Integration	436
Monitoring the API Connect cluster on Cloud Pak for Integration	436
Monitoring Postgres disk usage on Cloud Pak for Integration	438
Obtaining simple health check data of Developer Portal sites by using a REST API call	441
Logging	441
Gathering post-mortem logs	441
Changing logging levels	442
Troubleshooting	443
Troubleshooting on Kubernetes	443
Troubleshooting the management subsystem	443
Troubleshooting the management database	443
Increasing the PostgreSQL archive_timeout on Kubernetes	444
Recovering when disks are filled by the management database	444
Resizing a PersistentVolumeClaim for Postgres by volume expansion	445
Resizing a PersistentVolumeClaim for Postgres by moving content to a new PVC	446
Troubleshooting Analytics on Kubernetes	448
Analytics offload issues on Kubernetes	448
Empty Analytics display on Kubernetes	448
Analytics running out of disk space on Kubernetes	449
Troubleshooting on OpenShift and Cloud Pak for Integration	450
Power-cycling check on OpenShift and Cloud Pak for Integration	450
Troubleshooting the Management subsystem on OpenShift and Cloud Pak for Integration	450
Troubleshooting the management database on OpenShift and Cloud Pak for Integration	450
Increasing the PostgreSQL archive_timeout on OpenShift and Cloud Pak for Integration	451
Recovering on OpenShift and Cloud Pak for Integration when disks are filled by the management database	452
Resizing a PVC for Postgres on OpenShift by using volume expansion	452
Resizing a PVC for Postgres on OpenShift by moving content	453
Troubleshooting Analytics on OpenShift and Cloud Pak for Integration	455
Analytics offload issues on OpenShift and Cloud Pak for Integration	455

Empty Analytics display on OpenShift and Cloud Pak for Integration	455
Analytics running out of disk space on OpenShift and Cloud Pak for Integration	456
Uninstalling API Connect	457
Uninstalling API Connect subsystems	457
Removing Management subsystem postgres resources on Kubernetes	459
Removing API Connect	461
Removing postgres resources on Kubernetes after removing API Connect	462
Removing API Connect on OpenShift or Cloud Pak for Integration	464
Removing postgres resources on OpenShift and Cloud Pak for Integration	465
VMware	466
Planning your deployment	466
Deployment overview for endpoints and certificates	467
Planning your deployment topology and profiles	467
Enable JWT instead of mTLS	468
Firewall requirements on VMware	469
Firewall enabled ports for clustered OVA deployments	471
Load balancer ports for clustered OVA deployments	472
Load balancer configuration in a VMware deployment	472
A two data center deployment strategy on VMware	475
Planning your Analytics deployment	476
Planning the Analytics deployment profile	476
Planning to offload data to a third-party system	477
Planning to disable internal storage	477
Planning to modify your analytics data	477
Estimating internal storage space	477
Working with certificates	479
Certificate management: Read This First	480
Setting and managing certificates	480
Setting default certificates	481
Setting custom certificates	482
Replacing custom certificates	483
Setting common certificates	484
Setting the encryption-secret for the management database	485
Clearing certificates	485
Reference for certificates, commands, and validations	485
Certificate reference	485
Command reference	487
Validation reference	489
Tips and tricks for using APICUP	490
Installing API Connect	493
IBM API Connect Version 10 software product compatibility requirements	493
Requirements for initial deployment on VMware	494
API Connect licenses	496
What's new for v10 installation	496
First steps for deploying in a VMware environment	497
Signature verification by using PGP	93
Deploying the Management virtual appliance	499
Configuring the Management subsystem	499
Deploying the Management subsystem OVA file	503
Verify installation of the Management subsystem	504
Deploying the Analytics virtual appliance	506
Creating the analytics-extra-values file	506
Reference: Fields in analytics-extra-values	507
Enabling additional analytics features	507
Analytics persistent queue	508
Offloading data to a third-party system	508
Sample output plugins for offloading data	510
Providing a custom certificate for Analytics offload	511
Disabling internal storage	513
Modifying your analytics data	513
Sample filters for modifying analytics data	514
Deploying the Analytics subsystem	515
Configuring the Analytics subsystem	515
Deploying the Analytics subsystem OVA file	519
Verifying deployment of the Analytics subsystem	519
Deploying the Developer Portal virtual appliance	520
Configuring the Developer Portal subsystem	521
Deploying the Developer Portal subsystem OVA file	526
Verifying deployment of the Developer Portal subsystem	527
Configuring two NICs on the Developer Portal	528
Specifying a range of allowable client IP addresses for Developer Portal	529
Defining multiple portal endpoints for a VMware environment	529

Deploying DataPower Gateway virtual appliance	531
Installing DataPower Gateway	532
Configuring DataPower Gateway for API Connect	532
Configuring DataPower API Gateway	532
Configuring DataPower Gateway (v5 compatible)	536
Sample configuration for multiple peering objects on gateway services external to Kubernetes	538
Post-deployment steps	542
Installing a two data center deployment on VMware	543
Upgrading API Connect	549
Upgrade considerations on VMware	549
Additional considerations for upgrading analytics from a version prior to v10.0.5.0	552
Exporting analytics data from the UI	552
Exporting analytics with the REST API	553
Offloading analytics data	553
Update the analytics extra-values file	554
Preparing to upgrade on VMware	554
Upgrading a two data center deployment	555
Management subsystem two data center upgrade	556
Upgrading to the latest release on VMware	558
Control Plane files for earlier releases	564
Checksum values for earlier releases	567
Upgrading DataPower Gateway Service	567
Upgrading from 2018 on VMware	568
Upgrade from v2018: overview and considerations	568
Preparing the 2018 deployment for upgrade	570
Upgrading v2018 subsystems to v10	572
Upgrading Management subsystem from 2018 to v10	573
Upgrading Developer Portal from 2018 to v10	575
Upgrading Analytics from 2018 to v10	576
Enabling Analytics after upgrading	577
Troubleshooting upgrades on VMware	578
Maintaining API Connect	585
Advanced installation on VMware	586
Installing the IBM License Metric Tool on VMware	586
Installing the toolkit	320
Installing the Automated API behavior testing application	589
Advanced configuration on VMware	589
Adding a static route on a virtual machine	590
Adding disk space to a VMware appliance	590
Appliance boot mode configuration	591
Running commands at boot with iso-ignored boot mode	593
Changing deployment profiles on VMware	593
Changing the DNS server configuration on appliances	594
Configuring API Connect subsystems in a cluster	595
Configuring SSHD to limit access to your deployment	598
Enabling Developer Portal feature flags on VMware	599
Managing an appliance data disk	600
Replica override for microservices on VMware	601
Testing a new configuration against a current cluster	601
Use JWT security instead of mTLS between subsystems	602
Advanced configuration for management subsystem	602
Setting rate limits for public APIs on the management service	602
Configuring maximum size of client requests to the Management subsystem on VMware	604
Configuring monetization on VMware	605
Configuring use of an external NTP server	606
Overriding resources for Postgres components on VMware	607
Enabling UI feature flags on VMware	608
Backing up and restoring on VMware	609
Backing up and restoring the Management subsystem	609
Configuring backup settings during initial installation of the management subsystem	609
Verify configuration for s3 backup	612
Reconfiguring backup settings for the management subsystem	613
Backing up the management subsystem	616
Restoring the management subsystem	616
Troubleshooting management subsystem backups on VMware	618
Troubleshooting resiliency issues	618
Troubleshooting stanza-create job for S3 backup configuration	618
Backing up and restoring the Developer Portal	619
Configuring backup settings for the Developer Portal subsystem	620
Overview of the Developer Portal backup resources	359
Backing up the Developer Portal subsystem	623
Restoring the Developer Portal subsystem	624

Backing up and restoring the Analytics database on VMware	625
Configuring backup settings for Analytics	626
Running a backup of the Analytics database	627
Restoring the Analytics database	628
Disaster recovery	629
Preparing for a disaster	629
Preparing the management subsystem for disaster recovery on VMware	630
Preparing the Developer Portal subsystem for disaster recovery on VMware	630
Preparing the Analytics subsystem for disaster recovery on VMware	631
Recovering from a disaster	631
Recovering the Management subsystem on VMware	631
Recovering the Developer Portal subsystem on VMware	633
Recovering the Analytics subsystem on VMware	634
Using VM snapshots for infrastructure backup and disaster recovery	634
Disabling and re-enabling the Analytics subsystem on VMware	636
Maintaining a two data center deployment	637
Backup and restore considerations for a two data center deployment	637
Failure handling of a two data center deployment	638
Recovering from a failover of a two data center deployment	642
Removing a two data center deployment	643
Monitoring and health checks	644
Checking cluster health on VMware	644
Monitoring Postgres disk usage on VMware	645
Obtaining simple health check data of Developer Portal sites by using a REST API call	647
Monitoring the network with SNMP	647
Logging	648
Configuring remote logging for a VMware deployment	649
Gathering logs for a VMware environment	652
Changing logging levels	652
Troubleshooting	653
Recreating a calico pod that is in the CrashLoopBackOff state	654
Running a filesystem check on a VMware root partition	654
Troubleshooting the management database	654
Increasing the PostgreSQL archive_timeout on VMware	655
Dynamically re-registering and reconfiguring a Gateway service in a VMware deployment	656
Troubleshooting Analytics on VMware	657
Analytics offload issues on VMware	657
Empty Analytics display on VMware	657
Analytics running out of disk space on VMware	658
Migrating from API Connect v5 Public Cloud	659
Migration considerations	659
Preparing to migrate from v5 Public Cloud	662
Migrating data and Portal Delegated User Registries	665
Mapping v5 resource names to v10	668
Mapping v5 PDUR registries to v10	669
Migrating OAuth providers and OAuth secured APIs	670
Migrating v5 APIs to v10 API Gateway format	671
Configuration parameters for migrating v5 APIs	672
Reviewing and correcting migrated v5 APIs	676
Messages generated during conversion of APIs to DataPower API Gateway	676
Parse actions added to the API assembly after migration	677
Fine-tuning APIs after migration	677
Frequently asked questions about migrating APIs to API Gateway	678
Logging in to a management server with an OIDC registry	679
Migration troubleshooting	680
Migrating a Version 5 deployment	686
Migration overview	686
Migration steps	687
Migration steps with PDUR	688
Migration steps without PDUR	697
Logging in to a management server with an OIDC registry	679
Converting APIs to DataPower API Gateway	706
Configuring migration options for DataPower API Gateway	707
Reviewing APIs converted to DataPower API Gateway	711
Messages generated during conversion of APIs to DataPower API Gateway	712
Parse actions added to the API assembly after migration	748
Fine-tuning APIs after migration	748
Frequently asked questions about migrating to API Gateway	678
Migration mapping files	750
Migrating OAuth providers and OAuth secured APIs	670
Migrating token revocation information to DataPower API Gateway	754
Token migration errors	756
Migrating custom policies	756

Migrating gateway extensions	757
Migration considerations	659
Migration troubleshooting	680
Migrating from v10 to v10 on a different form factor	767
v10 migration overview	767
v10 migration requirements	769
v10 migration steps	771
Preparing the source system	772
Installing the target system	773
Restoring the target system	776
Verifying the target system	781
Migrating from OpenShift to Cloud Pak for Integration on the same cluster	781
v10 migration troubleshooting	783
Shutting down the Postgres database	784
Migrating from v2018 to v10 on a different form factor	786
Migration overview	786
Migration preparation	787
v2018 migration steps	789
Extracting the v2018 source data on Kubernetes and OpenShift	790
Extracting the v2018 source data on VMware	791
Installing the v10 target on Kubernetes	792
Installing the v10 target system on OpenShift and Cloud Pak for Integration	794
Installing the v10 target system on VMware	796
Restoring the v2018 data to the v10 target deployment	797
Verifying the v10 target deployment	800
Migrating additional v2018 subsystems: Staged migration	801
Troubleshooting the form-factor migration from v2018	801
Replacing passwords, keys, and certificates	802
Disabling backups before updating secrets and certificates	803
Management subsystem: updating the password, keys, and secrets	803
Changing the cloud administrator password	803
Changing the database encryption key	804
Changing the registration client_id and client_secret for applications	804
Gateway subsystem: updating the password and secret	806
Changing the Gateway administrator password	807
Changing the Gateway client_secret	807
Changing the secret for Gateway peering	807
Updating the TLS profile for Gateway	808
Portal subsystem: updating keys and secrets	809
Changing the secret for Portal data and system tools	809
Changing the Portal client_secret	810
Analytics subsystem: replacing the TLS profile and keys	810
Updating the TLS profile for Analytics	810
Updating the Analytics backup/restore secret	811
Renewing TLS certificates	811
Renewing certificates with cert-manager	812
Renewing the management CA with the ManagementSecretRotation CR	812
Renewing the portal CA with the PortalSecretRotation CR	813
Renewing the ingress-ca	814
Copying renewed ingress-ca to subsystems in different namespaces	815
Renewing the analytics CA	816
Renewing end-entity certificates	816
Monitoring cert-manager certificate renewal	816
Renewing custom certificates	817
List of Issuers, CA certificates, and secrets	818
List of ingress certificates	818
List of intra-subsystem certificates	820
V2018 upgrade: List of certificates to update manually	821
Renewing certificates on VMware	822
VMware: Renewing external certificates with apicup	823
Renewing certificates in a two data center deployment on Kubernetes and OpenShift	824
Management subsystem: restarting pods	826
Gateway subsystem: restarting pods	826
Portal subsystem: restarting pods	827
Analytics subsystem: restarting endpoints and pods	827
Restarting the Analytics ingestion endpoint	827
Restarting Analytics pods and data collection	828
Re-enabling scheduled backups	828
The API Connect operations tool: apicops	829
Configuring and managing your server environment	829
Cloud Manager configuration checklist	830
Activating your Cloud Manager user account	830
Accessing the Cloud Manager user interface	831
Defining your Cloud Manager topology	832

Creating an availability zone	833
Registering a gateway service	834
Registering an analytics service	836
Registering a portal service	838
Associating an analytics service with a gateway service	839
Setting visibility for a service	840
Managing authentication and security	841
User registries overview	841
Configuring an Authentication URL user registry	842
Configuring a shared custom user registry	844
Configuring an LDAP user registry in the Cloud Manager	849
Using the CLI to configure a shared LDAP user registry	851
Configuring a Local User Registry	858
Configuring an OIDC user registry	859
Setting visibility for a user registry	864
Deleting a user registry	864
Removing a user from a user registry	865
Removing yourself from a user registry	865
Removing another user from a user registry	866
TLS profiles overview	867
Viewing TLS Profiles, Keystores, and Truststores	867
Creating a TLS Server Profile	868
Creating a TLS Client Profile	869
Setting visibility for a TLS Client Profile	870
Defining elliptic curve cryptographic schemes for a TLS client profile	871
Creating a Keystore	872
Creating a Truststore	872
Viewing certificate details and adding certificates to a keystore or truststore	873
Generating a self-signed certificate using OpenSSL	874
Generating a PKCS#12 file for Certificate Authority	874
Binding a TLS server profile to a gateway service	875
Updating the PKCS#12 certificate for a TLS server profile	876
OAuth Provider overview	876
Configuring a native OAuth provider	877
Configuring basic settings for a native OAuth provider	879
Configuring scopes for a native OAuth provider	879
Configuring user security for a native OAuth provider	880
Configuring tokens for a native OAuth provider	881
Configuring token management and revocation for a native OAuth provider	882
Configuring introspection for a native OAuth provider	883
Configuring metadata for a native OAuth provider	884
Configuring the OIDC parameters for a native OAuth provider	884
Editing a native OAuth provider by using the API Editor	885
Configuring a third-party OAuth provider	886
Setting the visibility for OAuth providers	887
OAuth concepts for API Connect	887
OAuth user scenario	888
OAuth introspection for third-party OAuth providers	888
Oauth metadata URL and authentication URL	890
Scope	893
Tokens	897
Refresh tokens	897
OAuth revocation URL	897
Authenticating and authorizing through a redirect URL	899
Token management with the DataPower Gateway (v5 compatible)	901
Token management with the DataPower API Gateway	903
Authentication URL	905
Custom forms for user security	906
Creating a custom HTML login form for user security	906
Creating a custom HTML authorization form for user security	907
Securing an API with a JSON Web Token	908
Troubleshooting OAuth	908
Configuring the cloud settings	909
Change the name of your cloud	910
Configuring an email server for notifications	910
Setting up notifications	911
Customizing email notification templates	912
Configuring invitation timeouts	914
Configuring API Key timeouts	914
Using the CLI to modify cloud settings	914
Allowing the API key to be used for multiple logins	916
Configuring Base64 encoding for temporary tokens	916
Configuring the password-reset notification and timeout	916
Selecting user registries for Cloud Manager and API Manager	917

Setting a default user registry for Cloud Manager and API Manager	917
Configuring timeouts for access tokens and refresh tokens	918
Viewing platform and UI endpoints	919
Configuring the default gateway services for Catalogs	919
Managing the public/private key pairs for signing tokens	920
Configuring public/private key pairs for tokens	921
Administering provider organizations	922
Creating a provider organization	922
Editing a provider organization	924
Deleting a provider organization	924
Viewing current provider organizations	925
Sending messages to provider organization owners	925
Administering members and roles	926
Creating roles in the admin organization	927
Editing roles in the admin organization	927
Configuring LDAP group mappings on Cloud Manager user roles	928
Deleting roles in the admin organization	930
Viewing members and roles	930
Adding members to the admin organization	931
Assigning roles to members	932
Deleting a member	932
Role Defaults overview	932
Managing role defaults for provider organizations	933
Managing role defaults for consumer organizations	934
Monitoring the cloud	934
Accessing and configuring analytics	934
Counting total API calls for your Analytics service	935
Viewing API event data in API Connect	935
Cloud Manager analytics view	936
Including geoIP fields in analytics data	938
Configuring data retention and index rollover time periods	938
Configuring the Logstash persistent queue	939
Offloading analytics to third-party systems	939
Troubleshooting your analytics offload	939
Applying filters to incoming analytics data	940
Configuring audit logging to monitor user operations	940
Configuring the audit settings	943
Reviewing a gateway's processing status	945
Changing your Cloud Manager password and profile information	948
Resolving login problems by increasing HTTP header size	949
Onboarding a new admin for Cloud Pak for Integration	949
Extending the Gateway server behavior	950
Gateway extension guidelines - DataPower Gateway (v5 compatible)	951
Gateway extension guidelines - DataPower API Gateway	952
Gateway extensions manifest	954
v5 policy emulation limitations	955
Configuring your Gateway server extensions	956
Cloud Manager Tutorials	957
Tutorial: Configuring the Cloud	958
Tutorial: Creating a Provider Organization	966
Developing your APIs and applications	969
API development checklist	970
Working with the toolkit	970
Installing the toolkit	320
Logging in from API Designer	973
Setting the locale for API Designer	974
Switching clouds from API Designer	974
Working offline with API Designer	975
Gathering support information in API Designer	975
Using the developer toolkit command-line tool	977
Overview of the command-line tool	978
Logging in to a management server	980
Logging in to a management server with an OIDC registry	981
Cloud administration commands	982
API development and management commands	985
Creating APIs and applications	992
Publishing APIs and applications	993
Managing API Products	994
Working with Drafts	997
Reading input from the command line	997
Scripting with the toolkit commands	999
Working with OpenAPI extensions	1000
Managing platform REST API keys	1004
Searching for items in API Manager	1004
Working with API definitions	1005

Creating an API definition	1006
Creating a REST proxy API from a target service	1007
Creating a REST proxy API from an existing OpenAPI service	1008
Creating a REST proxy API from an existing WSDL service	1009
Creating a SOAP proxy API from an existing WSDL service	1010
Creating a new REST OpenAPI definition	1012
Adding a REST API by importing an OpenAPI definition file	1013
Adding a SOAP API by importing a zip file	1014
Creating a GraphQL proxy API	1015
Using the GraphQL schema editor	1017
Supporting introspection for a GraphQL API	1020
Enabling the GraphQL editor for a GraphQL API	1020
Enabling the cost endpoint for a GraphQL API	1021
Securing a GraphQL API by using a client ID	1021
Request mechanisms supported by GraphQL endpoints	1023
GraphQL custom schema directives and scalar types	1024
@remove directive	1024
GraphQL limitations	1025
Importing an API from the CP4I asset repository	1025
Editing an OpenAPI 2.0 API definition	1026
Specifying API metadata	1027
Specifying the host for an API	1027
Specifying the basepath for an API	1028
Specifying the schemes for an API	1028
Specifying the MIME types that an API consumes	1028
Specifying the MIME types that an API produces	1028
Enforcing security requirements on an API	1029
Defining security schemes	1030
Defining basic authentication security schemes	1030
Defining API key security schemes	1031
Client ID behavior in API key security definitions	1032
Defining OAuth2 security schemes	1033
Enabling CORS support for an API	1034
Specifying external documentation for an API	1035
Adding tags to an API	1035
Defining Paths for an API	1036
Defining parameters for a path	1036
Defining operations for a path	1037
Adding tags to an operation	1038
Specifying external documentation for an operation	1038
Specifying the MIME types that an API operation produces	1038
Specifying the MIME types that an API operation consumes	1039
Defining parameters for an operation	1039
Defining responses for an operation	1039
Specifying schemes for an operation	1039
Enforcing security requirements on an operation	1040
Defining schemas for an API	1041
Creating a schema	1041
Editing a schema	1042
Defining parameters for an API	1043
Creating a parameter	1044
Editing a parameter	1044
Defining responses for an API	1045
Creating a response	1045
Editing a response	1046
Specifying gateway and portal settings	1046
Defining target services for an API	1047
Setting API properties	1047
Defining Catalog specific property values	1048
Configuring application authentication for an API	1048
Configuring activity logging	1049
Configuring v5 compatibility options	1050
Including elements in your API assembly	1050
The assembly editor	1051
Adding elements to your assembly	1052
Handling errors in the assembly	1053
Specifying the gateway type for an API definition	1053
Converting an API definition for deployment to the DataPower API Gateway	1054
Modifying a GraphQL schema	1054
Validating the OpenAPI YAML source	1055
Editing an OpenAPI 3.0 API definition	1055
OpenAPI 3.0 support in IBM API Connect	1056

Specifying API metadata	1057
Defining servers for an API	1057
Enforcing security requirements on an API	1058
Enabling CORS support for an API	1059
Specifying external documentation for an API	1060
Adding tags to an API	1061
Defining Paths for an API	1061
Defining servers for a Path	1062
Defining parameters for a Path	1063
Defining operations for a Path	1063
Adding tags to an operation	1064
Specifying external documentation for an operation	1064
Defining parameters for an operation	1065
Defining the request body for an operation	1065
Defining responses for an operation	1066
Enforcing security requirements on an operation	1066
Defining servers for an operation	1067
Defining components for an API	1068
Defining schema components	1068
Creating a schema	1069
Editing a schema	1069
Defining response components	1071
Creating a response	1072
Editing a response	1072
Creating a content definition	1073
Editing a content definition	1074
Creating an encoding definition	1074
Editing an encoding definition	1075
Defining parameter components	1076
Creating a parameter	1076
Editing a parameter	1077
Defining example components	1078
Creating an example	1079
Editing an example	1079
Defining request body components	1080
Creating a request body	1080
Editing a request body	1081
Defining header components	1081
Creating a header	1082
Editing a header	1083
Defining security scheme components	1084
Defining basic authentication security scheme components	1084
Defining API key security scheme components	1085
Client ID behavior in API key security definitions	1032
Defining OAuth2 security scheme components	1087
Defining an HTTP bearer security scheme	1088
Defining link components	1089
Creating a link	1089
Editing a link	1090
Specifying gateway and portal settings	1091
Defining target services for an API	1091
Setting API properties	1092
Defining Catalog specific property values	1092
Configuring activity logging	1093
Including elements in your API assembly	1094
The assembly editor	1094
Adding elements to your assembly	1095
Handling errors in the assembly	1096
Validating the OpenAPI YAML source	1055
LDAP authentication	1098
Authentication URL user registry	1099
Activating an API	1100
Testing an API	1101
Locating API information on the Endpoints tab	1101
Using the Test tab to debug your API	1102
Preparing an API for debugging with the Test tab	1102
Specifying the testing preferences for an API	1103
Sending the API request	1104
Reviewing the API response and trace	1105
Testing a GraphQL API with the Test tab	1105
Testing an API with Automated API behavior testing	1106
Getting started	1107

Creating a test	1108
Modifying a test	1115
Running a test	1119
Publishing a test	1121
Tagging a test	1123
Variables	1126
Insights-Driven Test Generation	1128
Enabling Insights for Test Suites	1130
Generating Test Cases from Insights	1132
Watson Insights	1136
AutoTest Assist	1138
Creating a Profile	1139
Running a Profile	1143
Viewing the Report	1145
Profile Configuration	1149
Profile Security	1154
API Extensions	1156
Data Generation Rules	1159
Assertion components	1164
Assert Compares	1165
Assert Contains	1166
Assert Equals	1168
Assert Exists	1169
Assert Greater	1170
Assert In	1172
Assert Is	1173
Assert Less	1175
Assert Matches	1176
Set	1178
Security	1180
Storing sensitive data	1181
Additional tasks	1182
Template test from Specification Document	1183
Deleting a test suite	1185
Deleting a test	1186
Opening a test in the Editor	1187
API Hooks	1189
Creating An API Hook	1190
Creating API Keys and Secrets	1193
Using API Hooks	1196
List All Tests	1197
Run All Published Tests	1199
Run Published Tests By Tag	1202
Run Published Test By ID	1206
Automated API behavior testing tutorials	1209
Invoking test cases from Jenkins with JUnit result format	1210
Linking requests using variables	1216
Testing an API with the Policies editor	1219
Testing an API with the Local Test Environment	1220
Testing an API with the Explorer tab	1223
Staging an API	1224
Publishing an API	1226
Creating a new version of an API definition	1227
Updating an API definition from a file	1228
Updating a SOAP API	1228
Renaming an API definition	1229
Changing an API rate limit	1229
Adding an OpenAPI extension to an API	1229
Downloading an API definition	1230
Deleting an API definition	1230
Using an options file when importing a WSDL service	1231
Variable references in API Connect	1231
API properties	1234
API policies and logic constructs	1239
Built-in policies	1239
Activity Log	1242
Client Security	1243
Extract	1244
Constructing JSONata expressions to extract and transform data	1245
GatewayScript	1247
GatewayScript code examples	1248
Generate JWT	1250
Validate JWT	1252

GraphQL introspect	1253
Invoke	1254
Configuring the Invoke policy for DataPower API Gateway	1254
Configuring the Invoke policy for DataPower Gateway (v5 compatible)	1257
JSON to XML	1259
Configuring the JSON to XML policy for DataPower API Gateway	1260
Configuring the JSON to XML policy for DataPower Gateway (v5 compatible)	1261
Log	1262
Map	1263
The Map policy structure	1264
Configuring the Map policy in the user interface	1267
Map policy examples	1270
OAuth	1273
Example - using multiple OAuth policies in an OAuth provider assembly	1275
Parse	1278
Proxy	1280
Rate Limit	1283
Configuring a rate, burst, or count limit on the DataPower API Gateway	1284
Redaction - DataPower API Gateway	1286
Constructing JSONata expressions to redact fields	1286
Redaction - DataPower Gateway (v5 compatible)	1289
Constructing XPath expressions to redact fields	1290
Set Variable	1292
Configuring the Set Variable policy for DataPower API Gateway	1293
Configuring the Set Variable policy for DataPower Gateway (v5 compatible)	1294
User Security	1295
Validate - DataPower API Gateway	1297
Validate - DataPower Gateway (v5 compatible)	1300
Validate Username Token	1301
Websocket Upgrade	1302
Configuring the Websocket Upgrade policy for DataPower API Gateway	1302
Configuring the Websocket Upgrade policy for DataPower Gateway (v5 compatible)	1304
XML to JSON	1305
Configuring the XML to JSON policy for DataPower API Gateway	1306
Configuring the XML to JSON policy for DataPower Gateway (v5 compatible)	1307
XSLT	1307
Configuring the XSLT policy for DataPower API Gateway	1308
Configuring the XSLT policy for DataPower Gateway (v5 compatible)	1310
XSLT policy examples	1311
Implementation code examples	1314
Logic Constructs	1319
if	1320
operation-switch	1320
switch	1321
Using the switch policy condition editor	1321
throw	1325
Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway	1326
Manipulating attachments in a message object	1329
User-defined policies	1330
Tags in API Connect	1330
Working with Products	1331
Creating a draft Product	1332
Downloading a draft Product	1334
Importing a draft Product	1334
Searching for a draft Product	1335
Editing a draft Product	1336
Defining rate limits for an API operation	1339
Staging a draft Product	1340
Publishing a draft Product	1341
Creating a new version of your Product	1343
Specifying the gateway type for a Product	1344
Deleting a draft Product	1344
Organizing your Products into categories	1345
Creating and validating API and Product definitions by using the CLI	1345
Creating an OpenAPI definition file	1346
IBM extensions to the OpenAPI specification	1346
execute	1348
activity-log	1349
client-security	1350
extract	1351
gatewayscript	1352
graphql-introspect	1353
if	1354

invoke	1355
json-to-xml	1358
jwt-generate	1359
jwt-validate	1361
log	1362
map	1363
operation-switch	1367
oauth	1369
Example - using multiple OAuth policies in an OAuth provider assembly	1275
parse	1373
proxy	1376
ratelimit	1377
redact - DataPower API Gateway	1379
redact - DataPower Gateway (v5 compatible)	1380
set-variable	1382
switch	1383
Writing switch condition scripts - DataPower API Gateway	1384
Writing switch condition scripts - DataPower Gateway (v5 compatible)	1387
throw	1388
User-defined policies	1388
user-security	1389
validate - DataPower API Gateway	1391
validate - DataPower Gateway (v5 compatible)	1394
validate-username-token	1395
websocket-upgrade	1396
xml-to-json	1399
xslt	1399
catch	1402
properties	1402
catalogs	1402
activity-log	1403
Specifying a gateway type for an API definition	1404
Referring to an extension in an API definition	1404
Using \$ref to reuse code fragments in your OpenAPI files	1405
Creating a Product definition file	1406
Product definition schema	1407
Completing the information section of your Product description	1415
Specifying a gateway type for your Product	1416
Converting a Product YAML file to use DataPower API Gateway	1417
Specifying the visibility of your Product	1417
Referencing the APIs for your Product	1418
Describing Plans in your Product	1419
An example YAML representation of a Product	1421
Using x-ibm-languages to create multilingual API and Product documentation	1422
Using x-example to control the examples displayed in the Developer Portal	1425
Using x-embedded-doc to add additional documentation to products and APIs	1425
Using x-pathaliases to give consistent URLs for products and APIs	1428
Validating the YAML or JSON definition of an API or Product	1429
Creating and using API and Product definitions templates	1429
Working with global policies	1430
Customizing the preflow policies	1433
Reference	1434
API Connect context variables	1435
OAuth context variables	1440
Error cases supported by assembly catches	1442
API and Product definition template examples	1443
Template variables for API and Product definitions	1450
Managing your APIs	1451
API management checklist	1452
Activating your API Manager user account	1452
Accessing the API Manager user interface	1453
Searching for items in API Manager	1453
Working with Catalogs	1454
Creating and configuring Catalogs	1454
Configuring Product visibility in a Catalog or a Space	1459
Configuring sub paths for Developer Portal sites	1460
Managing Catalog membership	1461
Importing a user-defined policy into a Catalog	1462
Importing an OpenAPI extension into a Catalog	1462
Retaining Version 5 vanity endpoint behavior in a Catalog	1462
Using syndication in API Connect	1463
Enabling Spaces in a Catalog	1464
Creating, modifying, and deleting Spaces	1465
Working with Spaces	1466

Managing Products in a Space	1466
Managing subscription requests in a Space	1467
Managing application subscriptions in a Space	1467
Managing Space membership	1467
Managing user access in a Space	1468
Managing Gateways in a Space	1468
Working with Products	1469
The Product lifecycle	1469
Monetizing your Products	1470
Adding a billing integration resource	1471
Billing integration resource job queues	1472
Adding a billing integration resource to a Catalog	1472
Defining a Product with billing integration	1473
Managing your Products	1474
Considerations when changing a Product lifecycle with billing integration	1475
Publishing a Product	1476
Publishing a new Product	1477
Replacing a Product with another Product	1478
Superseding a Product with another Product	1479
Migrating application subscriptions to another Product	1480
Migrating application subscribers to new Product versions	1481
Changing the availability of a Product	1481
Deprecating a Product	1482
Retiring a Product	1483
Re-staging a Product	1484
Working with the subscriptions in a Product	1485
Updating the gateway services for a Product	1485
Removing a Product from a Catalog	1486
Approving Product lifecycle and subscription requests	1486
Working with APIs	1487
Working with Plans	1488
Working with consumer organizations	1488
Creating a consumer organization	1489
Editing a consumer organization	1490
Deleting a consumer organization	1490
Working with the applications in a consumer organization	1491
Working with the subscriptions in a consumer organization	1491
Working with consumer organization groups	1491
Sending messages to consumer organization owners	1492
Working with developer applications	1493
Working with application subscriptions	1493
Security and authentication	1494
Creating a TLS client profile	1494
Creating a Keystore	1495
Creating a Truststore	1496
Generating a PKCS#12 file for Certificate Authority	1497
Generating a self-signed certificate using OpenSSL	1498
Viewing certificate details and adding certificates to a keystore or truststore	1498
Defining elliptic curve cryptographic schemes for a TLS client profile	1499
Authenticating by using your enterprise user registry	1500
Working with user registries	1500
Creating an Authentication URL user registry	1501
Creating an organization-specific custom user registry	1502
Creating an LDAP user registry in API Manager	1505
Using the CLI to create an organization-specific LDAP user registry	1508
Creating a Local User Registry	1513
Creating an OIDC user registry	1514
Modifying the configuration details for a user registry	1520
Password lockout criteria	1520
Configuring a native OAuth provider	1520
Configuring basic settings for a native OAuth provider	1522
Configuring scopes for a native OAuth provider	1523
Configuring user security for a native OAuth provider	1524
Configuring tokens for a native OAuth provider	1524
Configuring token management and revocation for a native OAuth provider	1525
Configuring introspection for a native OAuth provider	1526
Configuring metadata for a native OAuth provider	1527
Configuring the OIDC parameters for a native OAuth provider	1527
Editing a native OAuth provider by using the API Editor	1528
Configuring a third-party OAuth provider	1529
OAuth concepts for API Connect	887
OAuth user scenario	888
OAuth introspection for third-party OAuth providers	888
OAuth external URL and authentication URL	890

Scope	893
Tokens	897
Refresh tokens	897
Token revocation	897
Authenticating and authorizing through a redirect URL	899
Token management with the DataPower Gateway (v5 compatible)	901
Token management with the DataPower API Gateway	903
Authentication URL user registry	905
Custom forms for user security	906
Creating a custom HTML login form for user security	906
Creating a custom HTML authorization form for user security	907
Securing an API with a JSON Web Token	908
Troubleshooting OAuth	908
API Analytics	1551
Catalogs, Spaces, and Analytics	1552
Accessing analytics	1552
Analytics dashboards	1553
Exporting API event data	1555
API event record fields	1556
Accessing analytics data with the toolkit CLI	1563
Example toolkit CLI operations	1564
Accessing analytics data with the REST API	1567
Modifying analytics API event data	1568
Administering user access	1571
Setting the expiration for invitations	1572
Configuring notifications	1572
Configuring sender details for email notifications	1574
Viewing your user information	1576
Adding provider organization users and assigning roles	1576
Creating custom roles	1577
Removing a user from a provider organization	1578
Configuring LDAP group mappings on API Manager user roles	1578
Changing your API Manager password	1580
Resolving login problems by increasing HTTP header size	1581
Reviewing a gateway's processing status	1581
Reference	1582
API gateway response codes	1582
Troubleshooting your billing configuration	1583
Authoring policies	1583
Authoring policies for the DataPower Gateway (v5 compatible)	1584
Describing your policy	1585
Creating a new user-defined policy	1587
Implementing your policy	1588
Packaging and importing your policies into IBM API Connect	1589
Reference	1590
Implementation code examples	1590
Authoring policies for the DataPower API Gateway	1595
Defining, packaging, and publishing a catalog-scoped policy for the API Gateway	1595
Structure of the catalog-scoped user-defined policy YAML file	1597
Defining, packaging, and publishing a global-scoped policy for the API Gateway	1599
Developer Portal: socialize your APIs	1601
Using the Developer Portal	1602
Logging in as a consumer to the Developer Portal by using the CLI	1602
Exploring APIs and Products in the Developer Portal	1603
How to use a URL to navigate directly to a Product or API in the Developer Portal	1603
Testing an API by using the Developer Portal test tool	1604
Calling an API	1605
Calling an API by using CORS	1606
Applications in the Developer Portal	1606
Registering an application	1606
Managing applications	1607
Editing an application	1608
Deleting an application	1608
Changing an application image	1608
Verifying an application client secret	1609
Analytics in the Developer Portal	1609
Application analytics in the Developer Portal	1609
Organization analytics in the Developer Portal	1609
Consumer organizations in the Developer Portal	1610
Adding a Consumer organization from within the Developer Portal	1610
Editing the name of a Consumer organization	1610
Switching Consumer organizations	1610
Adding users to a Consumer organization	1611
Removing a user from a Consumer organization	1611

Changing the ownership of a Consumer organization	1611
Deleting a Consumer organization	1613
User accounts, passwords, and support in the Developer Portal	1614
Creating a new developer account	1615
Changing your account settings	1615
Deleting your developer account	1615
Configuring the Developer Portal site	1616
Concepts in the Developer Portal	1617
Getting started configuring the Developer Portal site	1618
Appearance	1620
Controlling general appearance elements in the Developer Portal	1620
Changing the shortcut icon	1620
Changing the site email address	1621
Changing the site logo	1621
Changing the site name or site slogan	1621
Changing the visibility of site branding	1622
Adding custom JavaScript to a custom theme	1622
Applying a modified content type template	1622
Creating a sub-theme	1623
Installing additional themes	1626
Deleting themes	1627
Content	1627
Adding content elements in the Developer Portal	1627
Adding and configuring meta tags	1628
Disabling external search engine indexing	1629
Adding custom pages	1629
Integrating Twitter data into the social block	1629
Editing the social block	1630
Adding content in multiple languages	1630
Adding custom pages to APIs and Products	1631
Adding new frequently asked questions	1631
Applying an image to an API or Product	1632
Attaching a documentation file to APIs	1632
Configuring blogs	1633
Configuring the taxonomy menu block	1633
Customizing the URL alias for a specific API or Product page	1634
Embedding multimedia in site content	1634
Linking from one piece of site content to another	1634
Linking to social media sites	1635
Managing tags in the Developer Portal	1635
Tagging APIs based on their lifecycle phase	1635
Posting a poll	1636
Adding images to site content	1636
Configuring and restricting content in the Developer Portal	1637
Creating content types	1637
Adding fields to content types	1638
Configuring documentation upload limitations	1638
Configuring image upload limitations	1639
Configuring the appearance of ratings in the Developer Portal	1639
Customizing the privacy policy statement	1640
Customizing the terms of use statement	1640
Editing a site comment	1641
Editing sample content	1641
Editing tags for a specific item	1642
Turning content on or off in the Developer Portal	1642
Deleting a site comment	1642
Deleting blocks	1643
Disabling blogs	1643
Deleting forums from the front page	1644
Turning comments for specific content types on or off in the Developer Portal	1645
Turning comments off for an individual item	1645
Turning off ratings for specific content types in the Developer Portal	1646
Turning the ability to tag specific content types off in the Developer Portal	1646
Structure	1646
Configuring the front page	1647
Disabling blogs	1643
Deleting forums from the front page	1644
Adding a menu	1649
Adding and changing the blocks displayed on Developer Portal pages	1649
Changing the front page banner block	1650
Changing the front page Featured Content block	1651
Changing the items in the main menu	1652

Changing the ratings display	1653
Displaying APIs and Products in categories	1654
Enabling dynamic category creation	1654
Implementing an image carousel	1655
Making your application's image public	1656
Providing navigation by tag hierarchy	1656
Using the Views module in the Developer Portal	1657
Creating views in the Developer Portal	1657
Configuring the search results view in the Developer Portal	1658
Configuring the default number of items in a view list in the Developer Portal	1658
Configuration	1659
Configuring content moderation	1659
Editing, reviewing, and publishing moderated content	1661
General configuration tasks	1661
Adding a page load progress indicator	1662
Adding custom fields to user records	1663
Checking the site status report	1663
Clearing the server caches	1664
Configuring a proxy	1664
Configuring Stripe in the Developer Portal	1665
Configuring cron to run scheduled tasks	1666
Configuring search indexes and servers	1666
Configuring the date and time	1667
Configuring the notifications event log settings	1667
Configuring the settings of a Consumer organization	1667
Configuring the site default timezone	1668
Configuring the site error handling	1668
Configuring which buttons are displayed in the WYSIWYG rich text editor	1668
Configuring which languages are available	1669
Customizing user account settings	1669
Disabling languages	1670
Disabling test tool restrictions	1671
Enabling a cookie compliance banner	1671
Enabling code languages for code snippets	1672
Enabling code snippets for SOAP APIs	1672
Forcing new users to accept terms and conditions	1672
Hiding the admin registry on the login form	1673
Hiding the certificate in the header for APIs secured with mutual TLS	1674
Importing and exporting taxonomies	1675
Restricting access by IP address	1676
Restricting or preventing access from search engines	1678
Toggling the site in and out of maintenance mode	1679
Viewing available updates	1680
Managing Developer Portal security	1680
Configuring CAPTCHA	1681
Configuring reCAPTCHA	1681
Configuring session timeout and limit	1682
Configuring the timeout length for the password reset link	1682
Configuring your site password policy	1683
Disabling CORS warnings	1683
Disabling live testing of APIs	1684
How to enable penetration testing, and information about Developer Portal cookies	1684
How to manage IP security in the Developer Portal	1686
Managing banned IP addresses	1687
Using flood control for login security	1687
Login security	1687
Using the Security kit	1688
Using Honeypot for spam protection	1688
People	1689
Blocking and unblocking specific users	1689
Controlling access to Developer Portal content	1690
Creating a developer account to customize API properties	1690
Email Subscribers	1691
Emailing all subscribers	1691
Emailing api subscribers	1692
Emailing plan subscribers	1693
Emailing product subscribers	1693
Token reference	1694
How to style the Developer Portal emails	1696
Working with roles in the Developer Portal	1697
Creating administrator users for the Developer Portal	1698
Assigning users to a role	1698

Assigning permissions to a role	1699
Creating a new role	1699
Forums	1699
Creating a new forum	1700
Creating new forum containers	1700
Configuring the creation of new forums for each API	1700
Locking forum topics	1701
Marking content in a forum as sticky	1701
Removing forum posts	1701
Turning off forums in the Developer Portal	1702
Reports	1702
Extend	1702
Custom module development: an introduction to Drupal tools for PHP development	1703
Custom module development: background and prerequisites	1704
Custom module development: creating a module skeleton	1705
Custom module development: using hooks	1706
Installing custom modules	1708
Deleting custom modules	1709
Disabling modules	1709
Using Developer Portal metadata on API Manager	1710
Using a custom module for payment method creation	1710
Managing the Developer Portal by using the toolkit CLI	1710
Getting started with the Portal CLI commands	1711
Developer Portal CLI commands	1712
Using the api commands	1717
Using the apic-config commands	1718
Using the application commands	1718
Using the backups commands	1719
Using the consumer-org commands	1719
Using the content commands	1720
Using the custom-module commands	1721
Using the custom-theme commands	1722
Using the custom-translation commands	1722
Using the custom-webserver-page commands	1724
Using the drupal-config commands	1725
Using the drupal-state commands	1727
Using the entity commands	1728
Using the factory-reset command	1729
Using the forums commands	1729
Using the ip-security-enabled command	1730
Using the modules commands	1730
Using the php-memory commands	1731
Using the platforms commands	1732
Using the product commands	1732
Using the security command	1733
Using the service commands	1734
Using the service-ip-allowlist commands	1734
Using the site commands	1735
Using the site-config commands	1736
Using the sites commands	1737
Using the themes commands	1737
Using the twig commands	1738
Scenarios that use the Portal CLI	1739
Exporting and importing custom themes and site configuration	1739
Exporting and importing custom themes and site configuration by using the Portal Admin API	1743
How to enable and disable maintenance mode on your Developer Portal	1750
Troubleshooting guide for the Developer Portal	1751
Migrating your Developer Portal from Version 5 to Version 10	1755
Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10	1757
How to find your custom modules and themes on your Developer Portal	1757
Prepare your Developer Portal for the update for Drupal 10	1761
Upgrade your custom modules to be Drupal 10 and PHP 8.1 compliant	1764
Developer Portal tutorials	1767
Tutorial: Creating the Developer Portal	1767
Tutorial: Creating a private Developer Portal site	1771
Tutorial: Creating Accounts and Applications on the Developer Portal	1772
Tutorial: Creating a custom theme for the Developer Portal	1776
Tutorial: Using avatars	1779
Tutorial: Using image optimization	1782
Tutorial: Adding a Back to top button	1785
Tutorial: Creating a drop-down menu link	1786
Tutorial: Displaying blog posts on the front page	1790
Tutorial: Displaying tweets on the front page	1792
Tutorial: Changing the front page of your Developer Portal	1799

Tutorial: Grouping products by category	1805
Tutorial: Adding a field to the sign up form	1808
Tutorial: Adding validation to a field on the sign-up form	1810
Tutorial: Adding a field group to group fields in a content type	1812
Tutorial: Adding a custom block to the front page	1815
Tutorial: Adding a custom block to a page other than the front page	1818
Tutorial: Configuring the RobotsTxt file	1820
Tutorial: Configuring the SecurityTxt file	1821
Tutorials for using custom modules in the Developer Portal	1822
Tutorial: Changing the profile page to display firstname lastname, instead of username	1823
Tutorial: Adding validation to a field on the sign-up form	1810
Tutorial: Using a custom weighting sort order on the product list page	1826
Tutorial: Synchronizing application credentials from the Developer Portal with an external server	1828

Reference information

Toolkit command-line tool reference	1829
apic activations	1829
apic activations:clear	1830
apic activations:delete	1830
apic activations:get	1830
apic activations:list	1831
apic analytics	1831
apic analytics:create	1832
apic analytics-services	1832
apic analytics-services:clear	1832
apic analytics-services:create	1833
apic analytics-services:delete	1833
apic analytics-services:get	1834
apic analytics-services:list	1834
apic analytics-services:update	1834
apic api-keys	1835
apic api-keys:create	1835
apic api-keys:delete	1835
apic api-keys:get	1836
apic api-keys:list	1836
apic apis	1837
apic apis:clone	1837
apic apis:document	1838
apic apis:get	1838
apic apis:list	1838
apic apis:list-all	1839
apic apis:update	1839
apic apis:validate	1840
apic apis:wSDL	1840
apic apps	1840
apic apps:clear	1841
apic apps:create	1841
apic apps:delete	1842
apic apps:get	1842
apic apps:list	1843
apic apps:update	1843
apic associates	1843
apic associates:get	1844
apic associates:list	1844
apic availability-zones	1845
apic availability-zones:clear	1845
apic availability-zones:create	1845
apic availability-zones:delete	1846
apic availability-zones:get	1846
apic availability-zones:list	1847
apic availability-zones:update	1847
apic billings	1847
apic billings:clear	1848
apic billings:delete	1848
apic billings:get	1848
apic billings:list	1849
apic billings:update	1849
apic catalog-settings	1850
apic catalog-settings:get	1850
apic catalog-settings:update	1850
apic catalogs	1851
apic catalogs:clear	1851
apic catalogs:create	1851
apic catalogs:delete	1852

apic catalogs:email-to-owners	1852
apic catalogs:get	1852
apic catalogs:list	1853
apic catalogs:transfer-owner	1853
apic catalogs:update	1854
apic client-creds	1854
apic client-creds:clear	1854
apic client-creds:list	1855
apic client-creds:set	1855
apic cloud-emails	1855
apic cloud-emails:send-to-owners	1856
apic cloud-settings	1856
apic cloud-settings:about	1856
apic cloud-settings:audit-endpoint-test-connection	1857
apic cloud-settings:designer-credentials-list	1857
apic cloud-settings:get	1857
apic cloud-settings:info	1858
apic cloud-settings:mail-server-configured	1858
apic cloud-settings:oauth2-certs	1859
apic cloud-settings:toolkit-credentials-list	1859
apic cloud-settings:topology	1859
apic cloud-settings:update	1860
apic config	1860
apic config:clear	1860
apic config:delete	1861
apic config:get	1862
apic config:list	1862
apic config:set	1863
apic configured-api-user-registries	1864
apic configured-api-user-registries:clear	1864
apic configured-api-user-registries:create	1865
apic configured-api-user-registries:delete	1865
apic configured-api-user-registries:get	1865
apic configured-api-user-registries:list	1866
apic configured-billings	1866
apic configured-billings:clear	1867
apic configured-billings:create	1867
apic configured-billings:delete	1867
apic configured-billings:get	1868
apic configured-billings:list	1868
apic configured-catalog-user-registries	1869
apic configured-catalog-user-registries:create	1869
apic configured-catalog-user-registries:delete	1869
apic configured-catalog-user-registries:get	1870
apic configured-catalog-user-registries:list	1870
apic configured-catalog-user-registries:search	1871
apic configured-gateway-services	1871
apic configured-gateway-services:clear	1871
apic configured-gateway-services:create	1872
apic configured-gateway-services:delete	1872
apic configured-gateway-services:get	1873
apic configured-gateway-services:list	1873
apic configured-oauth-providers	1873
apic configured-oauth-providers:clear	1874
apic configured-oauth-providers:create	1874
apic configured-oauth-providers:delete	1875
apic configured-oauth-providers:get	1875
apic configured-oauth-providers:list	1875
apic configured-tls-client-profiles	1876
apic configured-tls-client-profiles:clear	1876
apic configured-tls-client-profiles:clear-all	1877
apic configured-tls-client-profiles:create	1877
apic configured-tls-client-profiles:delete	1877
apic configured-tls-client-profiles:get	1878
apic configured-tls-client-profiles:list	1878
apic configured-tls-client-profiles:list-all	1879
apic consumer-org-settings	1879
apic consumer-org-settings:delete	1879
apic consumer-org-settings:get	1880
apic consumer-org-settings:update	1880
apic consumer-orgs	1881
apic consumer-orgs:clear	1881
apic consumer-orgs:create	1881

apic consumer-orgs:delete	1882
apic consumer-orgs:get	1882
apic consumer-orgs:list	1883
apic consumer-orgs:transfer-owner	1883
apic consumer-orgs:update	1883
apic create	1884
apic create:api	1884
apic create:product	1885
apic credentials	1886
apic credentials:clear	1886
apic credentials:create	1887
apic credentials:delete	1887
apic credentials:get	1888
apic credentials:list	1888
apic credentials:reset-client-secret	1889
apic credentials:reset	1889
apic credentials:update	1889
apic credentials:verify-client-secret	1890
apic draft-apis	1890
apic draft-apis:clear-all	1891
apic draft-apis:clear	1891
apic draft-apis:clone	1891
apic draft-apis:create	1892
apic draft-apis:delete	1892
apic draft-apis:document	1893
apic draft-apis:get	1893
apic draft-apis:list	1893
apic draft-apis:list-all	1894
apic draft-apis:update	1894
apic draft-apis:update-wsdl	1894
apic draft-apis:validate	1895
apic draft-apis:wsdl	1895
apic draft-products	1896
apic draft-products:clear-all	1896
apic draft-products:clear	1896
apic draft-products:clone	1897
apic draft-products:create	1897
apic draft-products:delete	1897
apic draft-products:document	1898
apic draft-products:get	1898
apic draft-products:list-all	1899
apic draft-products:list	1899
apic draft-products:publish	1899
apic draft-products:update	1900
apic draft-products:validate	1900
apic drafts	1901
apic drafts:clear	1901
apic drafts:list	1901
apic entries	1902
apic entries:clear	1902
apic entries:create	1903
apic entries:delete	1903
apic entries:get	1903
apic entries:list	1904
apic entries:update	1904
apic extensions	1905
apic extensions:clear-all	1905
apic extensions:clear	1905
apic extensions:clone	1906
apic extensions:create	1906
apic extensions:delete	1907
apic extensions:document	1907
apic extensions:get	1908
apic extensions:list	1908
apic extensions:list-all	1908
apic extensions:update	1909
apic gateway-extensions	1909
apic gateway-extensions:create	1910
apic gateway-extensions:delete	1910
apic gateway-extensions:get	1910
apic gateway-extensions:implementation	1911
apic gateway-extensions:update	1911

apic gateway-services	1912
apic gateway-services:clear	1912
apic gateway-services:create	1912
apic gateway-services:delete	1913
apic gateway-services:get	1913
apic gateway-services:list	1914
apic gateway-services:reset-oauth-secret	1914
apic gateway-services:update	1914
apic global-policies	1915
apic global-policies:clear	1915
apic global-policies:clear-all	1916
apic global-policies:create	1916
apic global-policies:delete	1916
apic global-policies:document	1917
apic global-policies:get	1917
apic global-policies:list	1918
apic global-policies:list-all	1918
apic global-policies:update	1919
apic global-policy-errors	1919
apic global-policy-errors:create	1919
apic global-policy-errors:delete	1920
apic global-policy-errors:get	1920
apic global-policy-errors:update	1921
apic global-policy-posthooks	1921
apic global-policy-posthooks:create	1921
apic global-policy-posthooks:delete	1922
apic global-policy-posthooks:get	1922
apic global-policy-posthooks:update	1923
apic global-policy-prehooks	1923
apic global-policy-prehooks:create	1923
apic global-policy-prehooks:delete	1924
apic global-policy-prehooks:get	1924
apic global-policy-prehooks:update	1924
apic groups	1925
apic groups:clear	1925
apic groups:create	1926
apic groups:delete	1926
apic groups:get	1927
apic groups:list	1927
apic groups:update	1927
apic iam-apikey	1928
apic identity-providers	1928
apic identity-providers:list	1929
apic integrations	1929
apic integrations:clear	1929
apic integrations:create	1930
apic integrations:delete	1930
apic integrations:get	1931
apic integrations:list	1931
apic integrations:list-all	1931
apic integrations:update	1932
apic invitations	1932
apic invitations:clear	1932
apic invitations:create	1933
apic invitations:delete	1933
apic invitations:get	1934
apic invitations:list	1934
apic invitations:update	1935
apic jobs	1935
apic jobs:clear	1935
apic jobs:delete	1936
apic jobs:get	1936
apic jobs:list	1936
apic jobs:retry	1937
apic keystores	1937
apic keystores:clear	1938
apic keystores:create	1938
apic keystores:delete	1938
apic keystores:get	1939
apic keystores:list	1939
apic keystores:update	1939
apic licenses	1940
apic log-spec	1940

apic log-spec:get	1940
apic log-spec:update	1941
apic login	1941
apic logout	1942
apic mail-servers	1942
apic mail-servers:clear	1943
apic mail-servers:create	1943
apic mail-servers:delete	1943
apic mail-servers:get	1944
apic mail-servers:list	1944
apic mail-servers:test-connection	1944
apic mail-servers:update	1945
apic me	1945
apic me:change-password	1945
apic me:delete	1946
apic me:get	1946
apic me:update	1947
apic member-invitations	1947
apic member-invitations:clear	1947
apic member-invitations:create	1948
apic member-invitations:delete	1948
apic member-invitations:get	1949
apic member-invitations:list	1949
apic member-invitations:update	1949
apic members	1950
apic members:clear	1950
apic members:create	1951
apic members:delete	1951
apic members:get	1952
apic members:list	1952
apic members:update	1952
apic notification-languages	1953
apic notification-languages:get	1953
apic notification-languages:list	1954
apic notification-languages:update	1954
apic notification-styles	1955
apic notification-styles:get	1955
apic notification-styles:update	1955
apic notification-templates	1956
apic notification-templates:get	1956
apic notification-templates:list	1956
apic notification-templates:list-all	1957
apic notification-templates:update	1957
apic oauth-providers	1958
apic oauth-providers:clear	1958
apic oauth-providers:create	1958
apic oauth-providers:delete	1959
apic oauth-providers:get	1959
apic oauth-providers:list	1960
apic oauth-providers:update	1960
apic org-settings	1960
apic org-settings:get	1961
apic org-settings:update	1961
apic orgs	1961
apic orgs:clear	1962
apic orgs:create	1962
apic orgs:delete	1962
apic orgs:get	1963
apic orgs:list	1963
apic orgs:transfer-owner	1964
apic orgs:update	1964
apic payment-methods	1964
apic payment-methods:create	1965
apic payment-methods:delete	1965
apic payment-methods:get	1966
apic payment-methods:list	1966
apic payment-methods:update	1966
apic permissions	1967
apic permissions:get	1967
apic permissions:list	1968
apic permissions:list-all	1968
apic policies	1968
apic policies:clear	1969

apic policies:clear-all	1969
apic policies:clone	1970
apic policies:create	1970
apic policies:delete	1970
apic policies:document	1971
apic policies:get	1971
apic policies:implementation	1972
apic policies:list	1972
apic policies:list-all	1973
apic policies:update	1973
apic portal-services	1974
apic portal-services:clear	1974
apic portal-services:create	1974
apic portal-services:delete	1975
apic portal-services:get	1975
apic portal-services:list	1976
apic portal-services:update-credentials	1976
apic portal-services:update	1976
apic primary-events	1977
apic primary-events:get	1977
apic primary-events:list	1978
apic products	1978
apic products:clear	1979
apic products:clear-all	1979
apic products:clone	1979
apic products:delete	1980
apic products:document	1980
apic products:execute-migration-target	1981
apic products:get	1981
apic products:list	1981
apic products:list-all	1982
apic products:migrate-subscriptions	1982
apic products:publish	1983
apic products:replace	1983
apic products:set-migration-target	1983
apic products:supersede	1984
apic products:update	1984
apic products:validate	1985
apic properties	1985
apic properties:clear	1985
apic properties:create	1986
apic properties:delete	1986
apic properties:get	1987
apic properties:list	1987
apic properties:update	1987
apic registrations	1988
apic registrations:clear	1988
apic registrations:create	1989
apic registrations:delete	1989
apic registrations:get	1989
apic registrations:list	1990
apic registrations:update	1990
apic role-defaults	1990
apic role-defaults:clear	1991
apic role-defaults:create	1991
apic role-defaults:delete	1992
apic role-defaults:get	1992
apic role-defaults:list	1992
apic role-defaults:list-all	1993
apic role-defaults:update	1993
apic roles	1994
apic roles:clear	1994
apic roles:create	1994
apic roles:delete	1995
apic roles:get	1995
apic roles:list	1995
apic roles:update	1996
apic services	1996
apic services:clear	1997
apic services:clear-all	1997
apic services:create	1998
apic services:delete	1998

apic services:get	1998
apic services:list	1999
apic services:list-all	1999
apic services:update	2000
apic space-settings	2000
apic space-settings:get	2001
apic space-settings:update	2001
apic spaces	2001
apic spaces:clear	2002
apic spaces:create	2002
apic spaces:delete	2002
apic spaces:email-to-owners	2003
apic spaces:get	2003
apic spaces:list	2004
apic spaces:transfer-owner	2004
apic spaces:update	2004
apic subscriber-events	2005
apic subscriber-events:get	2005
apic subscriber-events:list	2006
apic subscriptions	2006
apic subscriptions:clear	2007
apic subscriptions:create	2007
apic subscriptions:delete	2007
apic subscriptions:get	2008
apic subscriptions:list	2008
apic subscriptions:update	2009
apic task-queues	2009
apic task-queues:get	2010
apic task-queues:list	2010
apic tasks	2010
apic tasks:get	2011
apic tasks:list	2011
apic tasks:update	2012
apic tls-client-profiles	2012
apic tls-client-profiles:clear	2013
apic tls-client-profiles:clear-all	2013
apic tls-client-profiles:create	2013
apic tls-client-profiles:delete	2014
apic tls-client-profiles:get	2014
apic tls-client-profiles:list	2014
apic tls-client-profiles:list-all	2015
apic tls-client-profiles:update	2015
apic tls-server-profiles	2016
apic tls-server-profiles:clear	2016
apic tls-server-profiles:clear-all	2016
apic tls-server-profiles:create	2017
apic tls-server-profiles:delete	2017
apic tls-server-profiles:get	2017
apic tls-server-profiles:list	2018
apic tls-server-profiles:list-all	2018
apic tls-server-profiles:update	2019
apic truststores	2019
apic truststores:clear	2019
apic truststores:create	2020
apic truststores:delete	2020
apic truststores:get	2020
apic truststores:list	2021
apic truststores:update	2021
apic user-registries	2022
apic user-registries:clear	2022
apic user-registries:create	2022
apic user-registries:delete	2023
apic user-registries:execute	2023
apic user-registries:get	2023
apic user-registries:list	2024
apic user-registries:search	2024
apic user-registries:test-connection	2025
apic user-registries:update	2025
apic user-registry-settings	2025
apic user-registry-settings:get	2026
apic user-registry-settings:update	2026
apic users	2026
apic users:clear	2027

apic users:create	2027
apic users:delete	2027
apic users:get	2028
apic users:list	2028
apic users:request-password-reset	2029
apic users:search-provider	2029
apic users:update	2029
apic validate	2030
apic version	2030
apic webhooks	2031
apic webhooks:get	2031
apic webhooks:list	2031
apic webhooks:update	2032
apic wsdl	2032
apic wsdl:introspect	2032
Portal command-line tool reference	2033
apic api	2033
apic api:add-attachment	2033
apic api:add-tag	2034
apic api:get	2034
apic api:get-document	2035
apic api:list	2035
apic api:set-icon	2035
apic apic-config	2036
apic apic-config:get	2036
apic application	2036
apic application:get	2037
apic application:list	2037
apic backups	2037
apic backups:list	2038
apic config	2038
apic config:clear	2038
apic config:delete	2039
apic config:get	2040
apic config:list	2040
apic config:set	2041
apic consumer-org	2042
apic consumer-org:get	2042
apic consumer-org:list	2042
apic content	2043
apic content:list	2043
apic content:list-types	2044
apic custom-module	2044
apic custom-module:create-export	2044
apic custom-module:create-import	2045
apic custom-module:delete-export	2045
apic custom-module:delete-import	2045
apic custom-module:get-export	2046
apic custom-module:get-export-status	2046
apic custom-module:get-import-status	2047
apic custom-theme	2047
apic custom-theme:create-export	2047
apic custom-theme:create-import	2048
apic custom-theme:delete-export	2048
apic custom-theme:delete-import	2048
apic custom-theme:get-export	2049
apic custom-theme:get-export-status	2049
apic custom-theme:get-import-status	2049
apic custom-translation	2050
apic custom-translation:create-export	2050
apic custom-translation:create-import	2051
apic custom-translation:delete-export	2051
apic custom-translation:delete-import	2051
apic custom-translation:get-export	2052
apic custom-translation:get-export-status	2052
apic custom-translation:get-import-status	2052
apic custom-webserver-page	2053
apic custom-webserver-page:delete	2053
apic custom-webserver-page:get	2053
apic custom-webserver-page:set	2054
apic drupal-config	2054
apic drupal-config:delete	2055
apic drupal-config:get	2055

apic drupal-config:list	2055
apic drupal-config:set	2056
apic drupal-state	2056
apic drupal-state:delete	2056
apic drupal-state:get	2057
apic drupal-state:set	2057
apic drupal-state	2056
apic drupal-state:delete	2056
apic drupal-state:get	2057
apic drupal-state:set	2057
apic entity	2059
apic entity:count	2059
apic factory-reset	2060
apic factory-reset:delete	2060
apic forums	2060
apic forums:disable	2061
apic forums:enable	2061
apic ip-security-enabled	2061
apic ip-security-enabled:update	2062
apic licenses	2062
apic modules	2062
apic modules:delete	2063
apic modules:disable	2063
apic modules:enable	2064
apic modules:list	2064
apic php-memory	2064
apic php-memory:list	2065
apic php-memory:update	2065
apic platforms	2065
apic platforms:list	2066
apic product	2066
apic product:add-attachment	2066
apic product:add-tag	2067
apic product:get	2067
apic product:get-document	2068
apic product:list	2068
apic product:set-icon	2068
apic security	2069
apic security:clear-bans	2069
apic service-ip-allowlist	2069
apic service-ip-allowlist:add	2070
apic service-ip-allowlist:delete	2070
apic service-ip-allowlist:list	2070
apic service-ip-allowlist:remove	2071
apic site	2071
apic site:cache-rebuild	2071
apic site:check	2072
apic site:login-link	2072
apic site:state	2073
apic site-config	2073
apic site-config:create-export	2073
apic site-config:create-import	2074
apic site-config:delete-export	2074
apic site-config:delete-import	2074
apic site-config:get-export	2075
apic site-config:get-export-status	2075
apic site-config:get-import-status	2075
apic sites	2076
apic sites:check	2076
apic sites:list	2077
apic themes	2077
apic themes:delete	2077
apic themes:disable	2078
apic themes:enable	2078
apic themes:list	2078
apic themes:set-default	2079
apic twig	2079
apic twig:debug-disable	2080
apic twig:debug-enable	2080
apic twig:debug-status	2080
apic version	2081
API Connect REST APIs	2081
Example: Using the platform REST APIs to publish a product containing a SOAP API	2081
Example Product file: globalweatherproduct_1.0.0.yaml	2083

Example API definition: globalweather_1.0.0.yaml	2083
Example WSDL file: globalweather.wsdl	2088
API Connect TLS certificates	2090
TLS certificates organized by usage	2094
Troubleshooting API Connect	2098
Unable to access Management UIs	2098
Unable to access the Portal sites	2099
Login failure on Management UIs	2100
Login failure on Portal sites	2100
Management operations not reflected on Gateway or Portal	2101
Advanced maintenance operations	2101
Management subsystem Postgres failover steps	2101

API Connect overview

IBM® API Connect is an integrated API management offering, with capabilities and tooling for all phases of the API lifecycle. IBM API Connect 10.0.5.4 is the latest follow-on release to IBM API Connect 10.0.5 Long Term Support (LTS) release. If you're looking for a different version of API Connect, use the Change version list in the navigation pane to select the version that you require.

Key steps of the API lifecycle include create, secure, manage, socialize, monetize, and analyze. IBM API Connect Version 10 delivers enhanced capabilities for the market-leading IBM API management solution. In addition to the ability to deploy in complex, multi-cloud topologies, this version provides enhanced experiences for developers and cloud administrators in your organization.

IBM API Connect has two main focuses: the first is providing best in class API Management tooling, and the second is having a cloud native solution. This allows users to create, manage, and secure applications that are deployed across a variety of on-premises and cloud environments.

The following table explains the key phases in the API lifecycle in more detail.

Table 1. Key phases of the API lifecycle

Lifecycle Phase	Description
Create	Develop and write API definitions from an API development environment, eventually bundling these APIs into consumable products, and deploying them to production environments. For tutorials, walk-throughs, and in-depth guides for building, testing, and deploying APIs and Products in API Connect, see Tutorials , and Developing your APIs and applications .
Secure	Leverage the best-in-class API Gateway, gateway policies, and more, to manage access to your APIs and back-end systems. To learn more about adding security to your API, see <i>Enforcing security requirements on an API</i> for the OpenAPI version that you are working with: <ul style="list-style-type: none">OpenAPI 2.0: Enforcing security requirements on an APIOpenAPI 3.0: Enforcing security requirements on an API To learn more about how to add API Gateway policies to your API, see API policies and logic constructs .
Manage	Governance structures are built in to the entire API lifecycle, from managing the view/edit permissions of APIs and Products being deployed, to managing what application developers can view and subscribe to when APIs are deployed. To understand and leverage API Connect management and governance controls along the API lifecycle, see Managing your APIs .
Socialize	API Connect comes with an advanced Developer Portal that streamlines the onboarding process of application developers, and can be completely customized to an organization's marketing standards. To understand more about using the Developer Portal, see Developer Portal: Socialize your APIs .
Monetize	Bundle APIs into Products and Plans with monetization schemes that are built into those Plans, thereby opening up digital revenue streams. To understand more about how to leverage monetization strategies in API Connect, see Monetizing your Products .
Analyze	Developers and Product Managers alike are given the tooling in API Connect to understand their API traffic patterns, latency, consumption, and more to make data driven insights into their API initiatives. To learn more about how to leverage API Connect analytics tooling, see API Analytics .

API Connect has four major components: API Manager, Analytics, Developer Portal, and Gateway. These four components can be deployed in a variety of hybrid and multi-cloud topologies. The API Connect infrastructure can either be deployed and managed by an IBM team in an IBM Cloud environment, or it can be deployed and managed by the customers in their own dedicated environment or third-party cloud. There is also the option for having hybrid scenarios, for example, with the API Connect Reserved Instance Offering, users are able to have their API Manager and Developer Portal running in the IBM Cloud, but then place remote gateways next to their back-end services.

Configuration of customer deployed API Connect clouds is done through the Cloud Manager. For in-depth guides and instructions see [Configuring and managing your server environment](#).

For information about the API Connect components that provide these capabilities, and details about the strategy for packaging and publishing APIs for use by API consumers, see:

- [Packaging strategy and terminology in API Connect](#)
- [API Connect components](#)

Note: For a comprehensive technical guide to best practices, considerations, and deployment options for API Connect, see the [API Connect v10.x Whitepaper](#).

- [What's new in the latest release \(Version 10.0.5.4\)](#)
Find out about the newest features and the latest updates in API Connect.
- [Language versions offered by API Connect](#)
API Connect is offered in several languages, as enabled by the operating platform on which the product is installed.
- [Known limitations](#)
This page describes the known limitations for API Connect 10.0.5.x.
- [API Connect concepts](#)
To help you get started, read about the API Connect concepts and obtain a high level understanding of the API management solution.
- [API Connect glossary](#)
The IBM API Connect and Cloud Manager glossary of terms and definitions.
- [Accessibility features for IBM API Connect](#)
Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.
- [Legal information](#)
Notices, and terms and conditions for information centers.
- [IBM API Connect Considerations for GDPR Readiness](#)
Information about features of IBM API Connect that you can configure, and aspects of the product's use, that you should consider to help your organization with GDPR readiness.
- [Essential reading](#)
These articles by IBM API Connect product specialists provide a wealth of supporting information on APIs and the API economy.

What's new in the latest release (Version 10.0.5.4)

Find out about the newest features and the latest updates in API Connect.

IBM® API Connect 10.0.5.4 is the latest follow-on release to IBM API Connect 10.0.5 Long Term Support (LTS) release. IBM API Connect 10.0.5.4 includes the features that were previously delivered in the releases of 10.0.2.0 CD through 10.0.5.3 LTS, along with the following enhancements.

If you are moving to Version 10 from Version 5, also see the [What's new in Version 2018](#) page for the additional new features since Version 5.

Note: You can access the latest files from [IBM Fix Central](#) by searching for the API Connect product and your installed version. Full installation files for IBM API Connect can be downloaded from [Passport Advantage](#). As enterprises continue their digital transformation agendas, IT leaders are looking to APIs as a means to interconnect digital solutions across their hybrid and multi-cloud landscape. This API-led innovation helps organizations accelerate the delivery of new business models and superior client experiences as they create, secure, manage, and socialize APIs across multi-cloud landscapes. IBM API Connect Version 10 can help increase productivity and improve outcomes for teams involved in API-led innovation efforts, including the following roles:

- [Developer Portal administrators](#): Socialize APIs across multi-cloud landscapes.
- [What's new for DevOps](#): Streamline operations across a multi-cloud and hybrid cloud landscape.

What's new for Developer Portal site administrators

New Developer Portal CLI **product** commands

The new **product** commands allow you to update the images, attachments, tags and categories for products on your Developer Portal. The following **product** commands are added:

- **product:add-attachment**
- **product:set-icon**
- **product:add-tag**

For more information about the **product** commands and how to use them, see [Using the product commands](#).

New Developer Portal CLI **api** commands

The new **api** commands allow you to update the images, attachments, tags, and categories for products on your Developer Portal. The following **api** commands are added:

- **api:add-attachment**
- **api:set-icon**
- **api:add-tag**

For more information about the **api** commands and how to use them, see [Using the api commands](#).

New Developer Portal CLI **site** command

The new **site:login-link** command is now available for use through the IBM API Connect toolkit CLI. By running this command, you receive a one-time login link for the Developer Portal site admin account.

For more information about the **site** commands and how to use them, see [Using the site commands](#).

New flag for the modules and themes commands in Developer Portal CLI

The new **--custom** flag added to your **modules:list** and **themes:list** commands helps you find what modules and themes have been installed on your Developer Portal site after an install.

For more information about the **--custom** flag, see [Using the themes commands](#) and [Using the modules commands](#).

New Developer Portal **content** commands

The new **content** commands allow you to list and expand your Developer Portal site content. The following **content** commands are added:

- **content:list-types**
Lists allowed content types on your site.
- **content:list**
Lists all the entities in your Developer Portal site for the given content type and bundle.

For more information about the **content** commands and how to use them, see [Using the content commands](#).

New Developer Portal CLI **service** command

The new **service:list** command allow you to list the Developer Portal service currently installed.

For more information about the **service** command and how to use it, see [Using the service commands](#).

What's new for DevOps

New analytics deployment profile

A new analytics profile is available on Kubernetes, OpenShift, and Cloud Pak for Integration: [n3xc4.m32](#). Use this profile instead of the existing [n3xc4.m16](#) profile if you have a high analytics load, since the 16 Mi of memory in the n3c4.m16 profile can be insufficient.

JWT security between gateway and analytics

Network communication between the gateway and analytics subsystems can now be secured with JWT instead of mTLS. For more information, see [Enable JWT security instead of mTLS](#).

Cert-manager upgraded to version 1.9.1

API Connect 10.0.5.4 uses cert-manager 1.9.1. If your environment requires a manual installation or upgrade of cert-manager, the instructions are included as part of the API Connect installation and upgrade procedures.

Updates in the API Connect Migration Utility (AMU)

The following updates are available in the AMU:

- A new `--drafts-only` option is added to the `archive:push` command that enables you to push only draft APIs and Products, and skip all published APIs and Products. For more information, see [Migration steps with PDUR](#) or [Migration steps without PDUR](#).
- A new `--allow-old-node` option is added to the `archive:port-to-apigw` command. This option allows you to use an older version of node 18 if required, but it is at your own risk. For more information, see [Migration troubleshooting](#).
- A new `--porting-suffix` option is added to the `archive:port-to-apigw` command that allows the overriding of the `-apigw` suffix for renamed artifacts. The porting suffix is now appended to the version of artifacts that have versions, such as APIs and Products. For more information, see [Configuring migration options for DataPower API Gateway](#).
- [What's new in Version 10.0.5.3](#)
This page details the enhancements in IBM API Connect Version 10.0.5.3 since the previous release.
- [What's new in Version 10.0.5.2](#)
This page details the enhancements in IBM API Connect Version **10.0.5.2** since the previous release.
- [What's new in Version 10.0.5.1](#)
This page details the enhancements in IBM API Connect Version **10.0.5.1** since the previous release.
- [What's new in Version 10.0.5.0](#)
This page details the enhancements in IBM API Connect Version 10.0.5.0.

What's new in Version 10.0.5.3

This page details the enhancements in IBM® API Connect Version 10.0.5.3 since the previous release.

Important: Note the following key changes in IBM API Connect Version 10.0.5.3:

- From API Connect 10.0.5.3, the Developer Portal, which is built on the open source Drupal content management system, will be upgraded from Drupal 9 to Drupal 10. This upgrade also requires PHP 8.1. The upgrade tooling will update your Developer Portal sites, however, if you have any custom modules or themes, it is your responsibility to ensure their compatibility with Drupal 10 and PHP 8.1 before the upgrade. For more information about the Drupal 10 upgrade and how to prepare for it, see [Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10](#).
- Sender details for email notifications for `consumer:app-suspended` and `consumer:app-reinstated` are updated to be consistent with the notifications in the consumer space. See [What's new for API product managers](#) for more details.

What's new for Developers

Added APIs to manipulate attachments in a message object

The message object provides APIs and properties to manipulate message attachments. For more information, see [Manipulating attachments in a message object](#).

Added the `JSON` custom scalar type for GraphQL APIs

You can use the `JSON` custom scalar type in GraphQL APIs to specify JSON payloads. For more information, see [GraphQL custom schema directives and scalar types](#).

Added the ability to specify strict UTF-8 encoding for JSON documents

When you configure a Parse policy, you can specify whether to enforce strict UTF-8 encoding throughout the entire JSON document. For more information, see [Parse](#).

Client credentials are shown in response to the `apps:create` command

If you're using the toolkit CLI to create consumer applications, the `apic apps:create` command now returns the client credentials for the application. For example, running the following command:

```
apic apps:create -s my_mgt_endpoint -o my_org -c my_catalog --consumer-org my_consumer-org tests/fixtures/refs/testapp.json
```

with the following payload:

```
{"title":"Jessica's Store","name":"store","redirect_endpoints":[]}
```

returns the following response:

```
store [state: enabled] https://example.com/api/apps/e6a4d5f1-bec0-416b-b1b7-738cc854c5b5/463d84ea-e70a-4338-beed-f3c2fe2547c0/423e0d4d-ea7c-4512-a5f6-afbd60bc3920/dab4a161-fdcd-455a-b5e7-c8447cc52105 client_id:'12340747f8bed76c4807988603bffa1f' client_secret: 'f123d1098ace3aa02d35d3f0da65a9f3'
```

For more information about how to run toolkit CLI commands, see [API development and management commands](#).

Updates to Automated API behavior testing OpenAPI 3.0 support

You can use OpenAPI 3.0 APIs in Automated API behavior testing. For more information, see [Testing an API with Automated API behavior testing](#).

What's new for API product managers

Sender details for email notifications for `consumer:app-suspended` and `consumer:app-reinstated` are updated to be consistent with the notifications in the consumer space

The sender address details for the following two email templates are now consistent with the other email notifications:

- `consumer:app-suspended`
- `consumer:app-reinstated`

Previously, `app-suspended` and `app-reinstated` notifications were always sent from the Cloud level sender name and address that's configured in the Cloud Manager. However, this configuration does not follow the established pattern for notifications, as documented in [Configuring sender details for email notifications](#). These two templates are now sent from the sender address that's configured at the Catalog level, if provided, or the Provider Organization level, or, if neither of those sender details are provided, the Cloud Manager level, consistent with existing behavior for other templates.

Specify the fields that are returned from analytics REST API and toolkit CLI event queries.

You can specify the API event record fields that you want returned from REST API and toolkit CLI event queries. Use the `--fields` argument with the toolkit CLI, and the `fields` query parameter with the REST API, for example:

```
/events?fields=bytes_received,bytes_sent
```

Analytics discover view in context filtering

While you're viewing the analytics API event data in the discover view, you can now quickly add and remove filters to update the event data displayed. For more information about filtering analytics UI data, see [Filtering displayed data](#).

Analytics dashboards enhanced with new charts

Three new charts are added to the analytics dashboards.

- Usage Dashboard: Top APIs by request size.
- Usage Dashboard: Top APIs by response size.
- Monitoring Latency Dashboard: Top APIs by response time.

New analytics dashboard: Gateway Operations

A new analytics dashboard is provided, which shows the API usage and latency per gateway. For more information about the analytics dashboards, see [Analytics dashboards](#).

Ability to save and share analytics queries

You can now save and share analytics queries created in the Cloud Manager and API Manager UIs. For more information about saving and sharing queries, see [API Manager UI analytics view](#).

Geographic IP data included in analytics API event data

The API event data now includes geographic information of the client IP address that called the API, and of the gateway itself. The geoIP feature is disabled by default. To use this feature, see [Including geoIP fields in analytics data](#).

What's new for Developer Portal site administrators

Change to the behavior of the approval process for onboarding to the Developer Portal

The way that the approval process works for onboarding to the Developer Portal has changed. In earlier releases, if Self service onboarding approval is enabled, and a user was approved to access a Developer Portal with a specific consumer organization, that user would not need to go through the approval process again if they created more consumer organizations for the same Developer Portal.

Now, Self service onboarding approval is honored whenever a consumer organization is created. So, even if a user is approved to access a Developer Portal with one consumer organization, if they then try to create another consumer organization in the same Developer Portal, their request will still go through the approval process.

For more information, see [Creating and configuring Catalogs](#).

Developer Portal upgrade to Drupal 10

From API Connect 10.0.5.3, the Developer Portal, which is built on the open source Drupal content management system, will be upgraded from Drupal 9 to Drupal 10. This upgrade also requires PHP 8.1. As a result, all existing Developer Portal sites will be upgraded to Drupal 10, and all new sites will be based on Drupal 10 as well. Ensure that all modules and themes used on your Developer Portal site are compatible with Drupal 10 and PHP 8.1. While core modules and themes will be automatically made compatible as part of the upgrade, any custom modules or themes, whether created in-house or contributed by the Drupal community, need to be verified for compatibility before the upgrade.

For more information about the Drupal 10 upgrade, and how to ensure compatibility, see [Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10](#).

Dynamic translations for API Connect entities

The latest Developer Portal upgrade contains a dynamic translations feature for the API Connect entities, including Blocks, Nodes, and Menu Links. Now, when you update the existing translations for your API Connect entities, the changes are automatically reflected across your sites without the need of re-translation. The dynamic translations feature also includes adding new language codes and their accompanying translations.

Note that the translation changes may take 5-10 minutes to appear on the site as the cache recycles.

What's new for DevOps

(Red Hat OpenShift) Change from cloudctl to ibm-pak for installing and upgrading air-gapped deployments

Previous versions of API Connect used IBM Cloud Pak CLI (cloudctl) as the mirroring tool for installing and upgrading in air-gapped environments. API Connect 10.0.5.3 uses IBM Catalog Management Plug-in for IBM Cloud Paks (ibm-pak) instead. The ibm-pak plug-in extends the Red Hat OpenShift CLI (oc command) capability and streamlines the process of delivering installation images to the IBM Cloud Pak in an air-gapped environment.

For information on installing and upgrading API Connect on Red Hat OpenShift in an air-gapped environment, see the [air-gap installation instructions](#) and the [air-gap upgrade instructions](#).

Daily Management subsystem local backups are taken by default

The Management subsystem now takes local backups by default. On new installations or upgrades where the user does not specify backup settings, management backups are taken daily at 01:00 UTC, and retained for 14 days. Local backups are set by default to prevent the situation where no backup settings cause continuous accumulation of WAL archives to local PVC.

Local backups are stored on the management subsystem, and so cannot be used for disaster recovery. Configure remote S3 or SFTP backups for disaster recovery.

For more information about management backups, see:

- [Configure local management backups on Kubernetes](#).
- [Configure local management backups on OpenShift](#).
- [Configure management backup settings on OVA](#).

(Kubernetes, Red Hat OpenShift) Improved platform health status monitoring using Kubernetes events

A new feature is enabled in the IBM **APIConnect** operator on Kubernetes and Red Hat OpenShift to generate CR events whenever there is a change in the top-level CR (APIConnectCluster) or subsystem CRs (**ManagementCluster**, **GatewayCluster**, **AnalyticsCluster**, **PortalCluster**) status conditions. Along with CR status conditions, you can monitor these events by using any third-party alerting system.

For more information, see [Monitoring platform health using Kubernetes events](#) or [Monitoring platform health on Red Hat OpenShift using Kubernetes events](#).

Red Hat OpenShift-based deployments now manage foundational services through the API Connect operands

In previous releases of API Connect, a deployment on Red Hat OpenShift (including as a capability in Cloud Pak for Integration) managed the use of IBM Cloud Pak foundational services through dependencies that are enforced by OLM at the operator level. Beginning with API Connect 10.0.5.3, the dependency on foundational services is no longer managed at the operator level by OLM, but is instead handled at the operand level by the API Connect operator. This change provides more flexibility going forward by allowing the APIC operator to enforce the dependency only on the operands that actually require it.

This change applies to new installations only. For more information, see the appropriate [installation instructions for your Red Hat OpenShift environment](#).

Configure analytics retention and rollover settings in the Cloud Manager UI

In previous releases, the only way to modify the retention and rollover settings was by using the toolkit CLI. It is now possible to configure these settings in the Cloud Manager UI. For more information, see [Configure data retention and index rollover time periods](#).

New message column in management and top-level get CR output

A new **MESSAGE** column is added to the output of `kubectl get <cr name>`

for the management CR. The **MESSAGE** column provides additional information on the state of the management subsystem. For example,

```
kubectl get mgmt -n <namespace>
```

NAME	READY	STATUS	VERSION	RECONCILED VERSION	MESSAGE	AGE
management	17/17	Running	10.0.5.3	10.0.5.3-0	Management is ready	2d22h

On Cloud Pak for Integration, and on Red Hat OpenShift, if you installed API Connect with the top-level CR, the new **MESSAGE** column is also shown for the top-level CR. For example,

```
oc get apiconnectcluster -n <namespace>
```

NAME	READY	STATUS	VERSION	RECONCILED VERSION	MESSAGE	AGE
small	7/7	Ready	10.0.5.3	10.0.5.3-16	API Connect cluster is ready	2d5h

Automatic verification and reporting of two data center disaster recovery management database replication

In two data center disaster recovery deployments, the management subsystem automatically monitors the replication status of the management databases, and reports any failures in the management CR status output. For example,

NAME	READY	STATUS	VERSION	RECONCILED VERSION	MESSAGE	AGE
management	18/18	Warning	10.0.5.3-0	10.0.5.3-0	Management is ready. HA Status Warning - see HAStatus in CR for details	8m59s

```
status:
  haStatus
  {
    $$$$$$
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication not working, install or upgrade in progress.",
    "reason": "na",
    "status": "True",
    "type": "Pending"
  }
}
```

Gateway CR property `validateApimClient` renamed to `mtlsValidateClient`

The `validateApimClient` property is renamed to `mtlsValidateClient` to match what is used in the other subsystems. `validateApimClient` is deprecated. When you upgrade to 10.0.5.3, `validateApimClient` is automatically changed to `mtlsValidateClient` in the gateway CR.

Use JWT instead of mTLS for some inter-subsystem communications

Most of the network interactions between API Connect subsystems use mTLS. The use of mTLS requires that any load-balancers that are located between subsystems are configured for TLS pass-through. If you want to use TLS termination at your load-balancers instead of pass-through on certain routes, you can disable mTLS and enable JWT security instead. For more information, see [Enable JWT security instead of mTLS](#).

If you are upgrading from an earlier release, to enable this feature see the following topics:

1. [Red Hat OpenShift: Additional post-upgrade steps for upgrading to 10.0.5.3 from earlier 10.0.5 release.](#)
2. [Kubernetes: Additional post-upgrade steps for upgrading to 10.0.5.3 from earlier 10.0.5 release.](#)
3. [OVA: Enable JWT security between subsystems.](#)

Non-OVA only: Use service (in-cluster) communication between subsystems, instead of external ingress/route based communication

If your subsystems are installed in the same cluster, then Kubernetes services can be used for inter-subsystem communication, instead of the default ingress/route based communication that requires external exposure of HTTPS endpoints. For more information, see [In-cluster service communication between subsystems](#).

Non-OVA only: New analytics database deployment option: Dedicated storage

Analytics now supports the OpenSearch dedicated storage type. Dedicated storage allows the management of the OpenSearch cluster, and the storage of analytics data to be separated into different pods. Using dedicated storage provides greater stability and allows some configuration changes to be made without downtime. Dedicated storage is available on three replica deployments only.

New analytics dashboard: Gateway Operations

A new analytics dashboard is provided, which shows the API usage and latency per gateway. For more information about the analytics dashboards, see [Cloud Manager UI analytics view](#).

New `customPlatformApiHostname` field available for Developer Portal restore

A new `customPlatformApiHostname` field is available to use when restoring the Developer Portal. The field allows you to provide a custom platform API hostname override for the Developer Portal `restore system` or `restore all` commands. If the `API` hostname of your management subsystem platform has changed since taking the backup you are restoring, then update it to the new platform API hostname of the management subsystem by using this override. For more information about restoring the Developer Portal, see:

- [Backing up and restoring the Developer Portal in a Kubernetes environment](#)
- [Restoring Developer Portal on OpenShift and Cloud Pak for Integration](#)
- [Restoring the Developer Portal subsystem](#)

New `portalBackupRecordDays` field available for Developer Portal backup settings

The `portalBackupRecordDays` setting denotes the number of days that backup records (backups that exist on the remote backup server) are shown when you list your remote backups. It is defaulted to 30 days. For more information about Developer Portal, see [Backing up and restoring the Developer Portal in a Kubernetes environment](#).

What's new for security practitioners

The MD5 certificate fingerprint format is no longer displayed

API Connect no longer calculates and displays the certificate fingerprint format of MD5. However, the SHA1 and SHA256 certificate fingerprint formats are displayed.

Two additional claims added to the JWKS URL endpoint on the management platform REST API

The following two claims are added to the JSON Web Key Set (JWKS) URL (`/api/cloud/oauth2/certs`) endpoint that is hosted on the REST API of the management platform.

- **iss**: describes the issuer of the token, which is IBM API Connect.
- **purpose**: describes what kind of API Connect token the key can be used to verify.

What's new in Version 10.0.5.2

This page details the enhancements in IBM® API Connect Version **10.0.5.2** since the previous release.

What's new for Developers

Added the **Date** custom scalar type for GraphQL APIs

You can use the **Date** custom scalar type in GraphQL APIs to define custom scalars that specify dates. For more information, see [Extensions to GraphQL](#).

Ability to set the API Designer locale

You can set a different locale than your current operating system for the API Designer application, by using the `--lang` argument when you start the application. For example, by using the `--lang=es` argument when you start API Designer, the language is set to Spanish. For more information, see [Setting the locale for API Designer](#).

Ability to switch directories in API Designer

You can switch to a different directory after logging in to the API Designer application UI, by clicking Switch directory on the menu bar in the header. For more information, see [Logging in from API Designer](#).

OpenAPI validation

You can now click Validate in the API Manager and API Designer API editors to validate your OpenAPI YAML source. If validation errors are found, you can click each error to navigate directly to the section of the API form that contains the invalid YAML. For more information, see [Validating the OpenAPI YAML source](#).

Support for multiple OAuth2 flows in OpenAPI 3.0 APIs

If you're creating or editing OpenAPI 3.0 APIs, you can now select multiple flows (grant types) for your OAuth2 security schemes. For more information about how to create OAuth2 security schemes, see [Defining OAuth2 security scheme components](#).

Create and manage API keys for platform APIs

Create API keys associated with your user account, and then use them to authenticate when you call the APIs provided by API Connect. For more information, see [Managing API keys](#).

What's new for API product managers

Analytics UI enhancements

The analytics UI now includes the following enhancements:

- All analytics views can be switched between local time and Coordinated Universal Time.
- Analytics charts can be switched between linear and logarithmic axes, selectable from the chart options menu.
- Analytics data can be shown going back to the start of your data retention period, not just the last 30 days.
- When you create and edit queries, the custom time range option now provides precision to the second, rather than the minute.
- Up to 200 events per page now viewable in the discover view tables.
- Updated columns in the discover view, providing clearer and more useful information.
- New query fields for `gateway_ip`, `client_ip`, `immediate_client_ip`. Queries can use CIDR notation to search by subnet.
- New refresh button on dashboards to update the data that is displayed, without the need to reload the browser.
- Ability to save and load analytics queries. Saved queries are stored in the analytics backup.

Analytics CLI and REST API enhancements

New filter operators for use with the CLI and REST API: `startswith`, `endswith`, `regex`, `notoneof`. For more information, see [Example toolkit CLI operations](#).

Additional variables for customizing the email notification templates

More variables are available to add to the email notification templates, so that you can define custom templates with more flexibility when addressing the user that's receiving the email. The following user variables are now available:

- `originator`
- `originatorFirstName`
- `originatorLastName`
- `originatorEmail`
- `username`
- `email`
- `firstName`
- `lastName`

For more information, see [Configuring notifications](#) in the API Manager, and [Customizing email notification templates](#) in the Cloud Manager.

What's new for API consumers

Consumer analytics redesign

The analytics dashboards in the Developer Portal are redesigned and enhanced. As well as viewing the standard analytics metrics for APIs at the application and organization level, API consumers can now view additional information, including response time, status codes, number of errors, and API call history.

What's new for Developer Portal administrators

Use the toolkit CLI to enable and disable maintenance mode on your Developer Portal site

You can now enable and disable maintenance mode on your Developer Portal site, by using the Portal commands in the toolkit CLI. For more information, see [How to enable and disable maintenance mode on your Developer Portal](#).

New **drupal-state** commands available in the Developer Portal command-line tool

The **drupal-state** commands enable you to store and retrieve information about the state of your Developer Portal system. For more information, see [Using the drupal-state commands](#).

What's new for DevOps

New requirement when installing or upgrading Developer Portal subsystem on VMware

If there are multiple Management subsystems in the same project, you must set the Developer Portal subsystem's **platform-api** and **consumer-api** certificates to match those used by the appropriate Management subsystem to ensure that the Developer Portal subsystem is correctly associated with that Management subsystem.

Instructions are included in the following topics:

- Installation on VMware: [Configuring the Developer Portal subsystem](#)
- Upgrade on VMware: [Upgrading to the latest release on VMware](#)

New dbClientPrimaryuser certificate

The Management component has a new certificate and secret called dbClientPrimaryuser, see: [Internal certificates](#).

If your deployment is installed on Kubernetes, and you use customized internal certificates, see the additional upgrade step: [Add db primary certificate](#).

If your deployment is installed on OpenShift, Cloud Pak for Integration, or OVA then no action is required.

Additional requirement for disaster recovery when Management subsystem uses S3 backups

If you installed API Connect on Red Hat OpenShift or Cloud Pak for Integration by using the top-level **apiconnectcluster** CR, you must annotate the CR as part of the disaster recovery process. The instructions for recovering from a disaster include a new step to guide you through the annotation procedure.

New deployment profiles with new CPU licensing limits

New deployment profiles are available, and CPU licensing limits updated. The **spec.license.metric** now correctly maps to the **productMetric** annotation for the containers that require license entitlement. See [CPU limits, Memory limits, and licensing](#). Some existing profiles are deprecated, see [API Connect deployment profiles for OpenShift and Cloud Pak for Integration](#).

Portal CR status message field

A new message field has been added to the output of **kubect1|oc get PortalCluster**. This new field provides portal status information, for example:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	MESSAGE	AGE
portal	3/4	Pending	10	10.0.5.2-0		Mounted certificates are not valid	8d
portal	4/4	Running	10	10.0.5.2-0		1 site(s) are not in sync with API Manager	8d
portal	4/4	Running	10	10.0.5.2-0		Ready for service registration	8d
portal	3/3	Running	10	10.0.5.2-0		Ready to create sites	22h
portal	3/3	Running	10	10.0.5.2-0		1 site is installing	22h
portal	3/3	Running	10	10.0.5.2-0		Serving 2 sites	22h
portal	3/3	Warning	10	10.0.5.2-0		Preparing for site upgrade(s)	22h
portal	3/3	Warning	10	10.0.5.2-0		Site Upgrades Executing: (0) Queued(1) Failed(0) Success (0)	22h

Manual portal backup and restore supports multiple portal clusters

Kubernetes, Cloud Pak for Integration, and OpenShift only: The portal backup and restore CRs now have a **portalCluster** property that specifies which portal cluster to apply the backup or restore to. For more information, see: [Backing up and restoring](#).

New API consumption dashboard

Monitor total API calls in your cloud with the new consumption dashboard available in the Cloud Manager UI, see [Cloud Manager UI analytics view](#).

Compatibility matrix for Kubernetes and OpenShift versions included in documentation

Tables that show which versions of IBM API Connect are compatible with which Kubernetes and OpenShift releases are now provided as a quick reference in the software product compatibility requirements page: [IBM API Connect Version 10 software product compatibility requirements](#).

Managing Developer Portal backup resources

If you are using remote backups, and there are a large number of rows in your **portalbackup** list that have a **CR TYPE** of **record**, there might be a slow down in system performance, including during Developer Portal upgrades. Although there is no upper limit to how many portal backups you can have, it is recommended that you prune your remote backups so that the number of rows does not become excessively large. One **portalbackup** row of type **record**, corresponds to one backup file on the configured remote backup server. For more information, see [Overview of the Developer Portal backup resources](#).

What's new for security practitioners

Migrate token revocation information to DataPower® API Gateway

You can use the Token Migration Utility (TMU) to migrate token revocation information from a v5 quota enforcement server (QES) to a gateway-peering cache in v10 DataPower API Gateway. For more information, see [Migrating token revocation information to DataPower API Gateway](#).

What's new in Version 10.0.5.1

This page details the enhancements in IBM® API Connect Version **10.0.5.1** since the previous release.

Version 10.0.5.1 product release

The following sections detail the changes and additions in version 10.0.5.1.

Attention:

API Connect 10.0.5.1 is now supported to run on VMware. For information about installing and upgrading on VMware, see the [VMware](#) section.

What's new for Developers

The API policy version must match the gateway policy version

The way that API Connect validates the API policy version has changed. In earlier releases, API Connect published the API if its policy version was considered to be compatible with the gateway's version. If no version was specified, the policy version defaulted to 1.0.0. However, the published API might then fail at the gateway if the API policy version didn't match the gateway policy version.

Now, API Connect mitigates the problem by validating the API policy version prior to publishing the API. This new behavior means that API Connect only publishes an API to the gateway if the API policy version exactly matches the version that is present on the gateway. In addition, if a policy version isn't specified, it no longer defaults to 1.0.0. If the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

Note that this change in behavior does not affect APIs that are already published.

For more information about adding policies to your assembly, see [Adding OpenAPI 2.0 elements to your assembly](#) or [Adding OpenAPI 3.0 elements to your assembly](#).

More flexibility when customizing the preflow policies

You can now use the `mode` property to control how the preflow policies are applied. For more information, see [Customizing the preflow policies](#).

What's new for API product managers

Analytics latency charts now show percentile levels

To assist in the investigation of latency anomalies, the latency charts now include percentile lines.

Advanced options now available on analytics REST API and toolkit CLI

Advanced query filters are now available when querying analytics data. See [Example toolkit CLI operations](#) and [Analytics REST API](#).

Configure the visibility of Products at a Catalog or Space level

You can now modify your Catalog settings to define a visibility property at a Catalog or Space level. This setting then becomes the maximum level of visibility that is allowed for any Product that is published to that Catalog or Space. It can also be used as the default visibility setting for that Catalog or Space. For more information, see [Configuring Product visibility in a Catalog](#).

Note that this is an additional feature that is disabled by default, so current Product visibility behavior is not affected. The new feature also doesn't affect any Products that are already published.

What's new for Developer Portal administrators

New commands are available in the Developer Portal command-line tool

The following commands for managing forums are now available in the Developer Portal command-line tool:

- **forums:disable** - Deletes any existing forums taxonomy terms, and then disables the Forum module for a given site.
- **forums:enable** - Enables the Forum module for a given site.

For more information, see [Using the forums commands](#).

What's new for DevOps

Upgrade now available for API Connect version 2018.4.1.20 to version 10.0.5.1

You can upgrade directly to 10.0.5.1 from 2018 FP20 by following the appropriate instructions for your platform:

- [Upgrading from v2018 on Kubernetes](#)
- [Upgrading from v2018 on OpenShift](#)
- [Upgrading from v2018 on VMware](#)

API Connect version 2018.4.1.20 to version 10.0.5.1 form factor migration

You can now migrate from version 2018.4.1.20 or later to a different form factor on version 10.0.5.1 or later. For more information, see: [Migrating from v2018 to v10 on a different form factor](#).

API Connect 10.0.5.1 uses a new version of cert-manager

- Kubernetes, VMware: cert-manager 1.7.1
There is an additional step to upgrade cert-manager [Upgrading subsystems on native Kubernetes](#). When you refer to cert-manager in your CRs or commands, the following values must be updated to reflect the newer version of cert-manager: In VMware, references to cert-manager in CRs are updated during the upgrade.
 - `cert-manager-0.10.1.yaml` becomes: `cert-manager-1.7.1.yaml`
 - `ingress-issuer-v1-alpha1.yaml` becomes: `ingress-issuer-v1.yaml`
 - `certmanager.k8s.io` becomes: `cert-manager.io`
- OpenShift, Cloud Pak for Integration: cert-manager 3.21 (provided by Foundational Services)
Upgrade of cert-manager is done automatically during the API Connect upgrade. However, when you refer to cert-manager in your CRs or commands, the following values must be updated to reflect the newer version of cert-manager:
 - `cert-manager-0.10.1.yaml` becomes: `cert-manager-3.21.yaml`
 - `ingress-issuer-v1-alpha1.yaml` becomes: `ingress-issuer-v1.yaml`

- `certmanager.k8s.io` becomes: `cert-manager.io`

New `apicup` commands for the rotation of secrets on the Developer Portal subsystem

The following commands are now available for the rotation of secrets on the Developer Portal subsystem on VMware:

- `list-rotate-secrets` - to obtain a list of any Developer Portal secret rotations that were created by using the `rotate-secrets` command.
- `delete-rotate-secret` - to delete any Developer Portal secret rotations that were created by using the `rotate-secrets` command.

For more information, see [Tips and tricks for using APICUP](#).

Two site High Availability (HA) feature renamed to two site DR (2DCDR)

This feature has been renamed to avoid confusion with production HA deployment profiles, and to recognize the manual intervention required to failover to the passive data center. The passive data center in a 2DCDR configuration has also been renamed to the warm-standby datacenter. See [Two data center deployment strategy](#).

Note: The terms passive and multiSiteHA are still used in the CR YAML files. This is a documentation change only, and there is no change in the behavior of 2DCDR.

New installation instructions for 2DCDR on Cloud Pak for Integration

The steps for installing API Connect on Cloud Pak for Integration in a 2DCDR configuration are now documented. See [Installing a two data center deployment on Cloud Pak for Integration](#).

New installation instructions for 2DCDR on OpenShift

2DCDR installation on OpenShift is now documented in a dedicated topic, instead of sharing the topic with Kubernetes 2DCDR install. See [Installing a two data center deployment on OpenShift](#).

Reorganization of Kubernetes, OpenShift, and Cloud Pak for Integration installation documentation

The installation documents for these platforms have been reorganized to provide a more sequential flow for the reader, and reduce the amount of non-applicable information presented to them; see [Deploying on Kubernetes](#) and [Deploying on OpenShift and Cloud Pak for Integration](#).

Mutual TLS support for Management to Gateway communication

For individual subsystem installations on OpenShift and Kubernetes, mutual TLS can be enabled with a new `validateApimClient` setting. See [Installing Gateway on Kubernetes](#), [Installing Gateway on OpenShift in a dedicated namespace](#), or [Installing Gateway on OpenShift in a shared namespace](#) depending on your deployment platform and topology.

What's new in Version 10.0.5.0

This page details the enhancements in IBM® API Connect Version 10.0.5.0.

If you are moving to Version 10 from Version 5, also see the [What's new in Version 2018](#) page for the additional new features since Version 5.

Note: You can access the latest files from IBM Fix Central. Full installation files for IBM API Connect can be downloaded from [Passport Advantage](#).

Version 10.0.5.0 product release

The following sections detail the changes and additions in version 10.0.5.0.

Attention:

IBM API Connect Version 10.0.5.0 requires IBM DataPower® Gateway Version 10.5. However, the DataPower firmware for the 10.5 LTS stream is supported on only the 8496 appliance, and the IBM DataPower Gateway for Docker. Consequently, the DataPower container for API Connect 10.0.5.0 is supported to run on only Kubernetes, OpenShift, and IBM Cloud Pak for Integration.

Although API Connect 10.0.5.0 is not supported to run on VMware, support for VMware is available from API Connect 10.0.5.1 and later.

What's new for Developers

OpenAPI 3 APIs with a response wildcard can be published but are disabled on the Gateway.

If an OpenAPI 3 API contains a response wildcard (which is not supported), publishing is disabled for that API. You must correct the problem before you can publish the API.

Increased OpenAPI 3.0 support

The following updates have been made to the Open API 3.0 support for APIs that are enforced by the DataPower API Gateway:

- All assembly policies are now supported, including the `validate` and `map` policies.
- The OAuth2 with multiple flows security scheme is now supported.

For more information, see [OpenAPI 3.0 support in IBM API Connect](#).

Added the Extract policy to transform content from the API context

The Extract policy transforms content from fields in the API context. You use JSONata expressions to specify the fields and how to transform the content. For more information, see [Extract](#).

What's new for API product managers

New publish validation option when publishing your products

You can now select from the following publish validation options when configuring Catalogs in the API Manager UI:

- Publishing APIs with empty paths is not allowed
- Validate custom policy schema in an assembly
- Validate the references inside the API and perform additional OpenAPI validations
- Publishing APIs with duplicate base paths is not allowed

To configure these options, in the Catalog that you want to work with click the Catalog settings tab, and then click Publish validations. Edit. For more information about configuring Catalogs, see [Creating and configuring Catalogs](#).

Override the timeouts set by consumer organization invitations

You can now override the consumer organization timeout with the timeout that is set in the Catalog. For more information, see [Creating and configuring Catalogs](#).

New analytics dashboards and REST API

New analytics dashboards are implemented in the API Manager UI, replacing the use of Kibana. There's also a new analytics REST API for accessing your analytics data. For more information, see [Accessing analytics in the API Manager UI](#).

Mixed-case resource names now supported

You can now create resource names with mixed-case text, including names for users, APIs, Products, Catalogs, organizations, and so on. However, you cannot have duplicate names of resources where the only difference is the case. For example, while **MyAPI** and **myapi** are both supported names, they cannot both exist in the same namespace. Also, note that Developer Portal endpoints must be only lower-case.

What's new for DevOps

Redesigned Analytics subsystem

- The Analytics subsystem now uses Opensearch instead of Elasticsearch, resulting in a different set of microservices; for detailed installation information, see [Installing the Analytics subsystem](#).

Attention:

The upgrade to v10.0.5.0 or later will result in the deletion of all existing analytics data. If you want to retain your analytics data then you must export it before upgrade. For instructions on exporting, see [Additional considerations for upgrade to v10.0.5](#). It is not possible to import analytics data from earlier releases to v10.0.5.0 or later.

- Logstash persistent queues replace the message queue feature.
- New analytics toolkit CLI commands and REST API operations, covering many administration operations including those previously done by running `curl_es` commands inside storage containers. `Curl_es` operations are no longer supported.
- Minor changes to the backup and restore process, which requires the recreation of the S3 backup secret after upgrade.
- A new analytics dashboard has been added to the Cloud Manager UI, allowing the administrator to view analytics API events for each analytics service that is configured.

Pre-upgrade health-check script improved

The `apicops` pre-upgrade script has been expanded to include additional checks of your deployment's health. Before upgrading to this release of API Connect, you should run the script and check the results as explained in the upgrade instructions for each platform.

Support for migrating billing data from v5 to v10

The API Connect Migration Utility (AMU) now supports migrating your billing data from v5 to v10:

- All Stripe billing integrations in v5 are migrated to v10 and applied automatically to all Catalogs.
- All products that use paid plans will have the plans migrated.
- Consumer payment methods that exist in the v5 Portal will be automatically created in v10.

After the migration is complete, you must configure the Stripe integrations in the Developer Portal sites; see [Configuring Stripe in the Developer Portal](#) for detailed instructions. For more information about migrating, see [Migrating a Version 5 deployment](#).

Ability to disable the Portal web endpoint check

When you create or register a Developer Portal service, the Portal subsystem checks that the Portal web endpoint is accessible. However sometimes, for example due to the complexity of public and private networks, the endpoint cannot be reached. In this instance, you can disable the Portal web endpoint check so that the Developer Portal service can be created successfully.

To disable the endpoint check, complete one of the following updates depending on your platform:

On Kubernetes, OpenShift, and IBM Cloud Pak for Integration

Add the following section to the Portal custom resource (CR) template:

```
spec:
  template:
    - containers:
      - env:
        - name: PORTAL_SKIP_WEB_ENDPOINT_VALIDATION
          value: "true"
        name: admin
        name: www
```

On VMware

In your `apicup` project, create a file called `ptl-extra-values.yaml` (or edit the file if one already exists), and add the following section:

```
spec:
  template:
    - containers:
      - env:
        - name: PORTAL_SKIP_WEB_ENDPOINT_VALIDATION
          value: "true"
        name: admin
        name: www
```

For more information, see [Registering a Portal service](#).

What's new for security practitioners

Configure Base64 encoding for temporary tokens

You can now enable or disable Base64 encoding for the temporary tokens that are created for invitations, and password reset tokens. For more information, see [Configuring Base64 encoding for temporary tokens](#).

Language versions offered by API Connect

API Connect is offered in several languages, as enabled by the operating platform on which the product is installed.

Note: When API Connect is deployed as a Cloud Pak for Integration capability, it is translated to the same set of languages as the Cloud Pak for Integration product.

- Brazilian Portuguese
- Chinese (Simplified)
- Chinese (Traditional)
- Czech
- Dutch
- English
- French
- German
- Italian
- Japanese
- Korean
- Polish
- Spanish
- Russian
- Turkish

Known limitations

This page describes the known limitations for API Connect 10.0.5.x.

Note: The limitations that are documented on this page will be removed as they are addressed in the IBM® API Connect product. For the most up-to-date list of product limitations, visit the English version of this page.

Attention: API Connect 10.0.5.0 is not supported to run on VMware. However, support for VMware is available in API Connect 10.0.5.1 and later.

Possible upgrade failure on VMware

During an upgrade to API Connect 10.0.5.3 or later on VMware, it is possible to encounter an issue where stanza create pods will fail, which causes other management services to be in **CrashLoopBackOff** state because postgres is still not up. It is also possible that the postgres pod is up but is stuck in a **0/1 Running** state. When this failure occurs, you cannot **exec** into the postgres pod and its logs display a message ending like the following example:

```
2023-04-28 11:11:49,258 INFO: Reaped pid=435,
```

You can confirm the issue by completing the following steps:

1. Determine which node the postgres pod is running on:

```
kubectl get pods -owide
```

2. On the node hosting the postgres pod, run the following command:

```
ctr -n k8s.io tasks ls | grep UNKNOWN
```

The issue is confirmed when the command returns more than one match, with one of the results being matched to the postgres container. If you encounter this issue, contact IBM Support for assistance.

New ibm-apiconnect-operator pod fails to start due existing terminating ibm-apiconnect-operator pod holding lock

The IBM APIConnect operator uses leader election (leader-for-life) to ensure that only one operator/controller pod is running/active at a time. This relies on Kubernetes garbage collection to clean up the lock when the old operator pod is deleted. The new ibm-apiconnect operator pod fails to start if an existing terminating ibm-apiconnect operator pod is still holding the lock.

If you suspect you are encountering this issue, verify it by running the following command:

```
kubectl -n <namespace> logs <new-ibm-apiconnect-pod-name> | grep "LockOwner"
```

If the lock is the problem, the log contains a message similar to the following example:

```
existing lock", "LockOwner": "ibm-apiconnect-xxxxxx held by terminating pod
```

Resolve the issue by completing one of the following options:

- Force delete the terminating pod by running the following command:

```
kubectl -n <namespace> delete pod <terminating-ibm-apiconnect-pod-name> --force --grace-period=0
```

Kubernetes deletes the lock when the pod is deleted.

- Remove the lock manually by running the following command:

```
kubectl delete cm ibm-apiconnect-lock
```

API Designer might hang while activating a large imported API

When you import a large API using API Designer and attempt to activate the API in the import wizard, the process might hang. If you encounter this issue, you can work around the problem by completing the following steps:

1. On your local file system, locate the autoprodukt file called API-NAME-auto-product_API-VERSION.yaml.
2. Delete the file.
3. In API Designer, edit the newly imported API and activate it by clicking the Online toggle.

In general, it is best practice to activate an API using the Online toggle or by publishing the API with the Publish option.

API Designer on Windows: APIs that use WSDL might encounter errors, or fail to activate, publish, or update.


If you activate, publish, or update a REST or SOAP API that uses a WSDL file, the operation might fail and never complete. Work around the issue by using the autopublish API feature in the API editor.

Permission restrictions in the API Designer UI

The API Designer UI currently has the following permission restrictions:

- Developers that have been given only View permissions cannot see the Test tab in the API editor. For developers to be able to see the Test tab, they must be given a different permission level. For information about the default permission levels available, see [API Connect user roles](#).
- Users with API-Drafts permissions, but who don't have any Sandbox Catalog permissions, cannot see the Test preferences in the Sandbox Catalog. For these users to be able to see the Test preferences, they must be given the Developer or Administrator role on the Sandbox Catalog.

Deleted security requirements might remain in the API source

Security requirements that are deleted from APIs in the API Designer and API Manager UIs, might still remain in the source. To work around this issue, click the Source icon  in the API editor, and manually remove the security requirements.

Sometimes the completion of an SFTP restore on the Management subsystem does not restore the actual data.

There is a known issue where sometimes a restore from the Management subsystem's SFTP backup succeeds but the data is not restored. If this happens, run the restore again.

Customized application view is reset to the default after upgrade if forums are also disabled

When upgrading API Connect version 10.0.5.0 or earlier to version 10.0.5.1 or later, if you have disabled forums on your Developer Portal site, and you have customized the application view, the customized application view will be reset to the default after the upgrade. To work around this issue, you can save the customized application view before running the upgrade, and then import the view into the upgraded Developer Portal.

To save the customized application view, run the following commands:


```
mkdir /tmp/tmp_config_<site-alias>
drush @<site-alias> config-get views.view.applications > /tmp/tmp_config_<site-alias>/views.view.applications.yaml
```

Where `<site-alias>` is the alias of your Developer Portal site.

To import the saved view after the upgrade, put the `views.view.applications.yaml` file inside a directory where it's the only file, and then run the following command:

```
drush @<site-alias> config-import --partial --source=/tmp/tmp_config_<site-alias>/
```

Cannot add comments to APIs by using the Source view in the API Designer and API Manager UIs

Comments cannot be added to APIs in the API Designer and API Manager UIs by clicking the Source icon , and using the hashtag symbol.

Changing Product visibility from Custom to Public doesn't automatically remove the Consumer organizations and groups

In the API Designer and API Manager UIs, changing Product visibility from Custom to Public doesn't automatically remove the consumer organizations and groups, so the Product publish will fail. To work around this issue, manually remove all of the consumer organizations and groups.

Stale cache can result in unexpected behavior in the Cloud Manager and API Manager UI

Having a stale cache in your browser can result in unexpected behavior in the Cloud Manager and API Manager UI, such as fetch errors, incorrect data being displayed, and blank pages. To work around this issue, particularly if you have just upgraded your deployment, complete the following actions:

- Reload the browser window.
- If there is still an issue, clear the browser cache and then log back in to the UI.
- Try using a private browser window.
- If possible, try a different browser type.

If issues persist, contact IBM Support.

The "Select compatible gateway services" option in Test Preferences causes a "404 POST undefined" error while testing the API

In the Test Preferences, Target gateway services setting, selecting Select compatible gateway services and choosing a specific gateway results in a "404 POST undefined" error when you test the API in the API Explorer or the Test tab.

Workaround: To avoid this problem, select the Use all compatible gateway services target gateway option instead.

Analytics command restrictions

The following `--mode analytics` commands work only when the flag `--return_format` is set to either `json` or `yaml`:

- `clustermgmt:catAllocation`
- `clustermgmt:catIndices`
- `clustermgmt:catNodes`
- `clustermgmt:catRecovery`
- `clustermgmt:catShards`
- `clustermgmt:catAliases`

The following commands don't currently work on the Toolkit CLI, as they return only `text/plain`:

- `clustermgmt:getNodesHotThreads`
- `clustermgmt:getNodesHotThreadsById`

Scale limits are used by default for the plan, collection, and operation rate and burst limits

Scale limits are used by default for the plan, collection, and operation rate and burst limits. Therefore, the following burst limit response headers are not used for the converted burst limits:


- `"X-BurstLimit-Limit"`
- `"X-BurstLimit-Remaining"`

Instead, the former burst limits are reported as the following rate limits in the headers:

- `"X-RateLimit-Limit"`
- `"X-RateLimit-Remaining"`

Note that this limitation does not apply to assembly burst limits.

Options menus in the Catalog might be hidden

In a Catalog, in any of the different tabs such as Consumers, or Subscriptions, when clicking on the options icon  the menu items might be hidden. To work around the issue, reload the page and the menu items will appear.

Override plan rate limits are not displayed in the Endpoint tab

Any override plan rate limits that have been added to your API for individual operations are not displayed in the API Endpoint tab in the UI. Only the plan rate limit is displayed.

You cannot install a "standalone" API Connect deployment on an OpenShift cluster where Cloud Pak for Integration is present

If an OpenShift cluster has Cloud Pak for Integration installed, then all API Connect deployments on that cluster are considered to be part of the Cloud Pak for Integration deployment, which will attempt to integrate with the standalone API Connect deployments.

Consumer rate limit notifications are not available in Version 10

The ability to configure notifications for applications so that API consumers are alerted when the usage of an API is nearing its rate limit is not available.

Cannot log in to API Manager using a Twitter OIDC if email is required

When the Email required option is enabled in the Twitter user registry as explained in [Configuring an OIDC user registry](#), then invited pOrg members and owners are unable to log in. The workaround is to leave this optional setting unselected.

Configuration update mistakes cause admission webhook errors in OpenShift and Cloud Pak for Integration

In OpenShift and Cloud Pak for Integration, configuration update errors can cause the `APIConnectCluster` instance to go into the Pending state with an underlying component throwing an admission webhook error.

```
oc get apiconnectcluster -n APIC_namespace
```

```
NAME      READY STATUS      VERSION      RECONCILED VERSION  AGE
m1        admission webhook "vmanagementcluster.kb.io" denied the request:
ManagementCluster.management.apiconnect.example.com "ml-mgmt" is invalid: [spec.databaseBackup.restartDB.accept: Invalid
value: false: databaseBackup protocol has changed (from ' ' to 'objstore') and requires database restart. accept restartDB
to restart database, spec.databaseBackup.restartDB.accept: Invalid value: false: databaseBackup configuration has changed
and requires database restart. accept restartDB to restart database] Pending 10.0.1.5-3388-eus 10.0.1.5-3388-eus
2d6h
```

The 'strict' SameSite cookie causes incorrect invitations to consumer organizations

Using the 'strict' SameSite cookie might cause invitation links from emails to send users to a registration page where they are asked to create a new consumer organization instead of joining the organization they were invited to.

The workaround is to use the 'Lax' SameSite Cookie attribute.

Performance impact when publishing products from API Designer

When you publish a product from API Designer and have 10 or more catalogs available, you might experience a performance degradation.

Limitations to the OpenAPI 3.0 support

IBM API Connect supports the OpenAPI 3.0 specification, with some limitations. For information about what is supported, see [OpenAPI 3.0 support](#).

A GraphQL API that contains a `graphql-input-type-cost` rate limit fails to publish

A GraphQL API created in releases earlier than IBM API Connect Version 10.0.3.0 might contain a `graphql-input-type-cost` rate limit, which is no longer supported. If you attempt to use the automatic activation mechanism to publish the API, or manually add the API to a Product and attempt to publish that Product, the publish operation will fail. You can resolve this problem in either of the following ways:

- Remove the rate limit definition from the OpenAPI source for the API. For example, if the source is in YAML format, remove the following lines:

```
- name: graphql-input-type-cost
  operation: consume
```

- Edit the source for the Product and define a `graphql-input-type-cost` rate limit in all of the Plans that include the API.
Note: You can edit only a manually created Product, not a Product that is generated by the automatic activation mechanism.

Analytics restore on OpenShift with `enableOverride: false` gets stuck in pending state

If you set `enableOverride: false` to avoid overwriting indices during a restore, the restore operation should fail when existing indices are encountered.

However, the operation does not fail, it instead gets stuck in the pending state.

When you configure the restore operation, set `enableOverride: true` but be aware that existing indices will be overwritten.

If you attempted a restore using `enableOverride: false` and the operation is stuck, then remove that job and start a new restore operation.

Options menu is missing from items on the Product list pages

If, in the API Manager user interface, you open the Product list page in either a Catalog or a Space, the options menu icon  might be missing from alongside all of the listed Products. To resolve this problem, reload the page.

Testing an API with the Policies editor doesn't work in the API Designer UI

The API Designer UI doesn't support testing an API with the Policies editor in the Gateway tab. Instead, you can test your API by using the Test tab or the Explorer tab.

Update WSDL option is not supported in the API Designer UI

The Update WSDL option to update the configuration of an existing SOAP API from a WSDL file or a .zip archive, is supported only in the API Manager UI, or by using the developer toolkit. This option is not supported in the API Designer UI.

Upgrading a 3 node profile to IBM API Connect 10.0.3.0 might result in some `portal-db/www` pods being stuck in the Pending state

IBM API Connect 10.0.3.0 introduces the pod anti-affinity required rule, meaning that in a 3 node profile deployment, all 3 `db` and `www` pods can run only if there are at least 3 running worker nodes. This rule can cause some upgrades to version 10.0.3.0 to become stuck in the Pending state. To resolve this issue, see one of the following workarounds:

- On a Kubernetes deployment: [Troubleshooting upgrades on Kubernetes](#)
- On an OpenShift deployment: [Troubleshooting installation and upgrade on OpenShift](#)
- On a VMware deployment: [Troubleshooting upgrades on VMware](#)

A unique email address is required for all Developer Portal accounts

Remember that every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a Developer Portal user account (and associated consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in to the Developer Portal: **A user already exists with this email address.**

For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries.

Configured user registries incorrectly allow the unsupported `external_group_mapping_enabled: true` option

API Connect does not support a configured user registry that is based on a registry where `external_group_mapping_enabled` is set to `true`. Under certain conditions you can create a configured user registry based on a user registry with the unsupported option; if this happens you will encounter errors while onboarding users.

You can create and update user registries with the `apim user-registries:create` and `apim user-registries:update` commands. Both commands accept the `external_group_mapping_enabled` option. If you then create a configured user registry with the `configured_catalog_user-registries:create` command, the results depend on how you set the `external_group_mapping_enabled` option in the user registry.

Suppose you create a user registry (for example, call it UR) and you want to create a configured user registry (call it CUR) based on UR. Consider the following scenarios:

1. You create UR with the `external_group_mapping_enabled: false`. In this case, you can create your CUR based on that registry.
2. You create UR with `external_group_mapping_enabled: true`. In this case, you cannot base a configured user registry on it, so the creation of CUR fails.
3. You create UR with `external_group_mapping_enabled: false` and create CUR based on it, as in scenario 1. However, you then update UR and set `external_group_mapping_enabled: true`. In this case, you have a problem.

The operation described in scenario 3 is not currently blocked, so you will successfully create an unsupported CUR and then run into errors when you attempt to onboard users. To resolve the issue, complete one of the following workarounds:

- Update UR (the base user registry) and set `external_group_mapping_enabled: false`.
- Leave UR with `external_group_mapping_enabled: true` but delete the configured registry (CUR) that is based on it.

A Product republish, with the Preserve Subscriptions option, fails after a consumer organization group has been removed from the visibility settings

If you remove a consumer organization group from the custom visibility settings for a Product, and that group contains a consumer organization with an application that is subscribed to the Product, an attempt to republish the Product with the Preserve Subscriptions option will fail even if that consumer organization is then added to the custom visibility settings individually.

For a secured GraphQL API, GraphQL subscriptions cannot be tested by using the Test tab in the user interfaces

For a GraphQL API that is secured by client ID, GraphQL subscriptions cannot be tested by using the Test tab in the API Designer or API Manager user interfaces. The API can still be published and used in production.

You can test GraphQL subscriptions in either of the following alternative ways:

- Remove client ID security from the API, for testing purposes, then use the Test tab.
- Use an external test tool.

A published API whose assembly contains a catch block with an invalid policy will now correctly fail to republish

Previously, policies in an assembly catch block were not validated, so if an incorrect policy configuration was coded in the OpenAPI source for an API, within an assembly catch block, the API would still publish successfully. Now, policies in an assembly catch block are validated, so such an API will fail to republish and will first need to be corrected.

An existing user cannot be added to a Space

If you attempt to add an existing user to a Space, the operation cannot be completed because the Create button is not enabled. Instead, use the invitation mechanism. For details, see [Managing Space membership](#).

Note: The invited user must use the Sign in option on the activation form, rather than completing the registration details and using Sign up.

Adding an `operation-switch` policy to an API assembly produces an incorrect warning message

If you add a Version 2.0.0 `operation-switch` policy to an API whose gateway type is DataPower® API Gateway, the API Setup page on the Design tab displays an incorrect warning stating that the policy is not valid for the gateway type. This warning message can be ignored.

Simultaneous editing of the same API by multiple users might result in changes being overwritten

If one user saves changes to an OpenAPI 3.0 API, another user who has the same API open in the API editor might have their changes overwritten.

An API developed for IBM API Connect Version 2018.4.1 that uses an `apim.getvariable('message.body')` function call fails in IBM API Connect Version 10

With the DataPower API Gateway in IBM API Connect Version 2018.4.1, the type of object returned, for XML payloads, by an

`apim.getvariable('context_name.body')` function call in a GatewayScript policy in an API assembly depends on how the variable is stored in the gateway context, as follows:

- If the variable is stored as a Buffer (because the variable data was written as an XML string), an XML Nodelist is returned provided that `contextname.headers.content-type` is an XML type.
- If the variable is stored as a parsed XML Document (because the variable data was written as parsed XML, either as an XML document or an XML Nodelist), an XML document is returned.

With the DataPower API Gateway in IBM API Connect Version 10 however, the same function call always returns an XML Nodelist provided that `contextname.headers.content-type` is an XML type. Therefore, such an API developed for Version 2018.4.1 will fail in Version 10 if it is configured to expect an XML document, and will need to be reconfigured appropriately.

The problem does not occur with such an API migrated from IBM API Connect Version 5 because, with that release,

`apim.getvariable('context_name.body')` also returns an XML Nodelist for XML payloads, nor does it occur if you are using the DataPower Gateway (v5 compatible).

`context_name` could be `message`, `request`, or the name of the output from an `invoke` policy.

Validate policy limitations when GraphQL response contains a GraphQL server error

When a GraphQL response contains a GraphQL server error and no data, the assembly Validate policy generates an error on the missing data and overwrites the payload. When the response contains partial data and an error, the assembly Validate action validates the data and overwrites the payload. To work around this limitation, use the condition `$not($exists(message.body.errors))` in an assembly switch condition to skip the assembly Validate policy when the response contains errors.

S3 backup type must be specified when configuring S3 backups on CP4I Platform Navigator or OpenShift Form UI

When you configure S3 backups for the Management subsystem, you must specify an S3 backup type of `ibm` or `aws`. The graphical user interfaces for CloudPak for Integration and for OpenShift incorrectly state the specification of S3 backup type is optional. In the CP4I Platform Navigator, or the OpenShift Form UI, when you configure Management System > Database Backup, be sure to select the S3 provider type menu and specify a supported value. For more information see [Configuring backup settings for fresh install of the Management subsystem](#) and the Database Backup table in the Management subsystem section of [IBM Cloud Pak for Integration](#).

A management backup remains in a `Pending` state

A management backup might remain in a `Pending` state for a long period of time, particularly after having upgraded the management subsystem. To resolve this problem, you must remove the pending backup and operation lock, and then re-run the backup CR. To remove the pending backup and operation lock, complete the following steps:

1. Verify the names of the pending backup and operation lock by running the following commands:

```
kubectl -n get mgmtb | grep Pending
```

```
kubectl -n get cm | grep oplock
```

2. Delete the pending backup by running the following command:

```
kubectl -n <namespace> delete mgmtb <name-of-backup>
```

3. Delete the operation lock by running the following command:

```
kubectl -n <namespace> delete cm <name-of-lock>
```

GraphQL API test fails when the generated example query contains variable definitions

If, when testing a GraphQL API in the Test tab, you use the Example dropdown menu to generate a Small, Medium, or Large query, and the generated query contains variable definitions, those variables are not added to the HTTP request, which then fails with the following error:

```
Parser failed with error: syntax error, variables type is neither a JSON object nor null.
```

To resolve this problem, modify the query to remove the variable definitions before re-sending the HTTP request.

Changes to parameters in an API definition aren't reflected on the Test tab

If you add or remove parameters in an API definition on the Design tab and then open the Test tab, the changes might not be reflected immediately. If you are using the API Manager user interface, either restart the browser or clear the browser cache and cookies, if you are using the API Designer user interface, restart the application. Then reopen the Test tab in the API definition.

An OpenAPI definition that contains regular expression syntax fails validation

IBM API Connect supports the [GO regular expression syntax](#). When you import an OpenAPI definition into the API Designer or API Manager user interfaces, or validate one with the `apic validate`, the validation will fail if the OpenAPI source contains unsupported regular expression syntax, with errors that include `Does not match format 'regex'`; for example:

```
- Must validate at least one schema (anyOf) (context:
(root).paths./example/types.post.parameters.0.schema.properties.items, line: 0, col: 0)
- Must validate one and only one schema (oneOf) (context: (root).paths./example/types.post.parameters.0, line: 46164, col:
21)
- paths./example/types.post.parameters.0.schema.properties.items.properties.pattern Does not match format 'regex'
(context: (root).paths./example/types.post.parameters.0.schema.properties.items.properties.pattern, line: 0, col: 0)
```

During migration, the `archive:unpack` fails to find an OAuth provider

When migrating API Connect to Version 10, the `archive:unpack` command might generate a warning like the following:

```
WARN[2020-06-04T09:29:54-04:00] Leaving a placeholder OAuth Provider which must be replaced in draft API <api name>:
Unable to find an OAuth Provider to extract from Token URL or Authorization URL <api token/authorization URL> with
basePath </base path> and scopes <scope list>.
```

A possible cause is that an OAuth provider matches the API path but is missing the necessary scopes. To resolve this problem, complete the following steps:

1. Update the API definition in the `cloud/provider-orgs/<provider_organization_name>/drafts` folder to replace the `x-ibm-oauth-provider` value 'placeholder' with `name://../<oauth_provider>`.
2. Update the API definition in the `cloud/provider-orgs/<provider_organization_name>/catalogs/<catalog_name>/products` folder to replace the `x-ibm-oauth-provider` value 'placeholder' with `name://../<oauth_provider>`.
3. Update the OAuth Provider definition in the `cloud/provider-orgs/<provider_organization_name>/oauth-providers` folder to add the extra scopes from the API definition.

In addition, if the OAuth Provider references a user-registry in a `user-security` section, and this user-registry is not specified in the `cloud/provider-orgs/<provider_organization_name>/catalogs/<catalog_name>/configured-api-user-registries` folder, then add an entry of the following form to ensure that the user-registry is pushed to the Catalog:

```
api_version: 2.0.0
name: name://../<user_registries/<user_registry_name>/user-registry.yml
type: configured api user registry
user_registry_url: file://../<user_registries/<user_registry_name>/user-registry.yml
```

The migration of a user registry fails

When migrating API Connect to Version 10, a user registry might fail to be pushed to the API Manager with errors like the following:

```
ErrorLog2020-06-02T08:35:37-04:00.log:ERRO[2020-06-02T08:36:07-04:00] unable to create User registry <user registry name>
ErrorLog2020-06-02T08:35:37-04:00.log:ERRO[2020-06-02T08:36:07-04:00] orgldap: An error occurred communicating with the
ldap server at
'ldaps://<ldap-address>' (error: 'Error: self signed certificate in certificate chain').
```


To work around this problem, for any artifacts, such as APIs or OAuth providers, that reference this user registry in their User Security section, replace the user registry with one that has been successfully migrated.

The migration of an OAuth provider push fails due to a missing TLS client profile ID

When migrating API Connect to Version 10, an OAuth Provider might fail to be pushed to the API Manager with an error like the following:

```
unable to create Oauth provider <provider name>
<provider name>: The expected path parameter 'tls-client-profile-id' is missing from the request URL.
```

and subsequently an error like the following:

```
unable to create Configured oauth provider <provider name>
<provider name>: The 'oauth_provider_url' property value '/api/orgs/<org>/oauth-providers/<provider name>' does not refer to a known Oauth Provider.
```

This is caused by an OAuth provider containing a custom HTML form in one or more User Security sections that specifies an `ei-custom-form-tls-client-profile` or an `az-custom-form-tls-client-profile`.

To resolve this problem, use the default TLS client profile by specifying a value of `' '`.

When scaling down a Kubernetes cluster of DataPower gateways, the cluster might enter an unrecoverable state if the peering primary instance is lost

If you are scaling down a Kubernetes cluster of DataPower gateways, you must first complete the following steps to ensure that the peering primary instance is not lost:

1. List the pods in the cluster; for example:

```
$ kubectl get pods -n apiconnect --selector
app.kubernetes.io/component=datapower,crd.apiconnect.ibm.com/instance=gwv6
NAME      READY   STATUS    RESTARTS   AGE
gwv6-0    1/1     Running   0           3h31m
gwv6-1    1/1     Running   0           3h32m
gwv6-2    1/1     Running   0           3h34m
```

2. Enter the DataPower admin CLI for gateway 0, by entering the following command and logging in at the prompt; for example:

```
$ kubectl attach -n apiconnect -it gwv6-0
```

3. Switch to the API Connect domain and enter config mode:

```
idg# switch apiconnect
idg[apiconnect]# config
Global mode
```

4. Show the gateway peering status; for example:

```
idg[apiconnect](config)# show gateway-peering-status
```

Address	Configuration name	Pending updates	Replication offset	Link status	Primary
1.2.3.1	gwd	0	18331896	ok	no
1.2.3.1	rate-limit	0	3091219	ok	no
1.2.3.1	subs	0	3150127	ok	no
1.2.3.1	tms	0	3063575	ok	no
1.2.3.2	gwd	0	18330948	ok	no
1.2.3.2	rate-limit	0	3091067	ok	no
1.2.3.2	subs	0	3149975	ok	no
1.2.3.2	tms	0	3063257	ok	no
1.2.3.3	gwd	0	18330948	ok	yes
1.2.3.3	rate-limit	0	3091067	ok	yes
1.2.3.3	subs	0	3149975	ok	yes
1.2.3.3	tms	0	3063257	ok	yes

5. For each of the configuration names, set gateway 0 to be the peering primary instance:

```
idg[apiconnect](config)# gateway-peering-switch-primary gwd
The instance is now the primary in the peer group.
idg[apiconnect](config)# gateway-peering-switch-primary rate-limit
The instance is now the primary in the peer group.
idg[apiconnect](config)# gateway-peering-switch-primary subs
The instance is now the primary in the peer group.
idg[apiconnect](config)# gateway-peering-switch-primary tms
The instance is now the primary in the peer group.
```

6. Show the gateway peering status to confirm the changes; for example:

```
idg[apiconnect](config)# show gateway-peering-status
```

Address	Configuration name	Pending updates	Replication offset	Link status	Primary
1.2.3.1	gwd	0	19067953	ok	yes
1.2.3.1	rate-limit	0	3218172	ok	yes
1.2.3.1	subs	0	3277032	ok	yes
1.2.3.1	tms	0	3189160	ok	yes
1.2.3.2	gwd	0	19067953	ok	no
1.2.3.2	rate-limit	0	3218172	ok	no
1.2.3.2	subs	0	3277200	ok	no
1.2.3.2	tms	0	3189160	ok	no
1.2.3.3	gwd	0	19069241	ok	no
1.2.3.3	rate-limit	0	3218172	ok	no
1.2.3.3	subs	0	3277508	ok	no
1.2.3.3	tms	0	3189312	ok	no

Incorrect UI behavior if API changes are not saved before creating a new API version

If you make changes to an API definition and then attempt to create a new API version without first saving your changes, you are not prompted to save your changes until after you have completed the new version creation operation. If you click OK in the prompt, your new version is created but your original changes are lost; to create your new version and retain your original changes, you should click Cancel in the prompt, then click Save to save your original changes.

Truststore and keystore settings in a shared TLS client profile are not reflected in API Manager

If you create a shared TLS client profile in the Cloud Manager user interface and assign a truststore and a keystore, these truststore and keystore settings are not reflected in that TLS client profile in the API Manager user interface and cannot be set there. Furthermore, the TLS client profile list views display the value shared for the truststore and keystore rather than the correct values. To workaround this problem, create the truststore, keystore, and TLS client profile separately for each provider organization in the API Manager user interface, then enable the TLS client profile in your Catalogs as required.

Login to the Cloud Manager or API Manager user interfaces might fail with error 431 if the browser has a large number of cookies

Attempts to login to the Cloud Manager or API Manager user interfaces might fail if the HTTP header or cookie size is larger than 16 KB, a limitation that is imposed for security reasons. To resolve this problem, either clear the browser cache and cookies, or open a private window, then retry.

The `apic cloud-settings:topology` command doesn't return a topology object

If you use the `apic`

`cloud-settings:topology` command without a `--format` parameter, it returns only the string `CloudTopology` rather than the cloud topology. To retrieve the topology object, include the `--format` parameter, with a value of either `yaml` or `json` as required.

The `apic cloud-settings:mail-server-configured` command response is incorrect

If you use the `apic`

`cloud-settings:mail-server-configured` without a `--format` parameter, it returns only the string `MailServerConfigured` rather than a boolean value to indicate whether an email server has been configured in the cloud settings. To retrieve the correct response, include the `--format` parameter, with a value of either `yaml` or `json` as required.

API Designer user interface might not open correctly if there are badly formed OpenAPI YAML files in the startup directory

If, when launching the API Designer user interface, the startup directory that you select contains one or more badly formed OpenAPI YAML files, the interface might fail to load correctly, with an error such as the following:

```
Cannot read property 'type' of null
```

To resolve this problem, remove any invalid OpenAPI YAML files from the startup directory, then restart the API Designer user interface.

Pagination setting is global across the API Connect user interfaces

If you set the Items per page value on any page in the Cloud Manager or API Manager user interface, that setting is then applied to all pages in both user interfaces in the same browser session. If you want to set the value separately for a specific page, open it in a private browser window. Such a setting in a private browser window is specific to that window and is lost when the window is closed.

Refresh tokens are not supported in the API Manager or API Designer user interfaces with an OIDC registry

The use of refresh tokens is supported in API Manager, API Designer, the Developer Portal, and the toolkit CLI, provided that the user logs in with a non-OIDC Provider user registry. If the registry type used for user login is OIDC, then refresh tokens are not supported in API Manager or API Designer. For more information, see [Specifying cloud settings for refresh tokens](#).

Setting resource limits in Kubernetes can cause containers to behave oddly at run time

In Kubernetes, you can set resource limits for each container. However, limiting CPU means that the container processes might slow down if they try to use too much. Limiting memory means that the Kubernetes code kills processes that try to allocate memory that would take the container over its limit. These limits can result in odd behavior such as the containers continually trying to start a daemon process, such as `nginx` or `node`, and exiting with success immediately.

Field validation for the Client security policy is incorrect

When configuring a Client Security policy in an API assembly in the API Designer or API Manager user interfaces, there is the following incorrect validation behavior:

- The ID Name field is required, but the API definition can be saved without entering a value in the field.
- The Secret Name field is required only when the Secret Required option is selected, but the user interface indicates that the Secret Name field is required regardless. Furthermore, when the field is required, the API definition can be saved without entering a value.
- If the Authenticate Client Method is set to Third party, the User Registry Name field is required, but the API definition can be saved without entering a value in this field.

Requests that take longer than the value of the `nginx` timeout cause a 409 error

Requests that take longer than the value of the `nginx` timeout can cause a 409 error, such as:

```
Another request is operating on portal-subsystem-03729230-4df7-4419-94c9-e7691b616382 with the same name, please try again.
```

The error occurs because the ingress controller is sending a repeat request when the previous one timed out. Your request will still continue processing in the background.

Adding members to a provider organization might fail

When adding members to a provider organization, the Cloud Manager user interface lists the current owner and members as well. If existing members are selected, the add action will fail.

Upgrading with new certificates breaks analytics ingestion

If the analytics subsystem certificates are modified after installation (by using the `apicup certs set` command, followed by the `apicup install` command to deploy the change to the cluster), the analytics ingestion ingress will no longer be recognized by the Gateway. The analytics service must be re-registered in the Cloud Manager interface so that the new certificates are recognized.

Users are not logged out of the user interfaces after tokens expire

If an API Manager or Cloud Manager user interface session times out, the user does not get redirected to the login page and might see an error. To redirect to the login page, refresh the browser window.

User interfaces are not supported in Microsoft Edge

The API Manager and Cloud Manager user interfaces are not supported in the Microsoft Edge web browser. To work in the user interfaces, use a different browser.

New gateways cannot connect to existing gateways after the gateway peering certificates are changed

Changing the gateway peering SSL certificates after gateway installation causes new gateways to be unable to connect to existing gateways, resulting in an outage. A full restart of the gateway StatefulSet (by deleting all the StatefulSet pods) is required for the gateways to come back online.

Changes to an OAuth provider are not reflected in the consuming APIs

If you modify the endpoints, scopes, or grants for an OAuth provider, the APIs that consume that OAuth provider will be affected. You might need to update the consuming APIs accordingly.

No option to bulk delete APIs or Products

In the API Designer and API Manager user interfaces, there is currently no option to delete multiple APIs or Products in a single operation; in the user interfaces, APIs and Products must be deleted individually. However, you can bulk delete APIs and Products by using the REST API or CLI interfaces.

Cannot publish an imported API that was generated with IBM API Connect Version 5.0 LoopBack®

If you import an API YAML file that was generated with IBM API Connect Version 5.0 LoopBack, attempts to publish the API fail. Before publishing, remove the catalogs section, and the invoke policy named `tls-invoke-profile`, from the YAML source of the API definition; for example:

```
catalogs:
  apic-dev:
    properties:
      runtime-url: $(TARGET_URL)
  sb:
    properties:
      runtime-url: 'http://localhost:4001'
```

Cannot publish an API that has duplicate security definition entries

The API Designer and API Manager user interfaces allow you to add duplicate security definitions to an API. However, attempts to publish the API will fail with OpenAPI validation errors. Ensure that the security definitions in an API are unique.

Kubernetes might excessively evict pods

In Kubernetes, if insufficient memory, CPU, or storage resources are provided then pods are evicted randomly. To avoid this problem, allocate sufficient resources to the pods. In addition, there are known issues with log rotation in Kubernetes that are documented here:

<https://github.com/kubernetes/kubernetes/issues/59902>. The lack of log rotation can cause logs to grow, limited only by the amount of storage on the node, eventually causing pods to restart. Therefore ensure that appropriate log levels are configured for all API Connect components.

You may also follow the instructions described here <https://success.docker.com/article/how-to-setup-log-rotation-post-installation> to set the maximum size of the log before it is rolled (max-size) and the maximum number of log files that can be present (max-files). More information about the Docker JSON File logging driver is available here: <https://docs.docker.com/config/containers/logging/json-file/>.

Logging in to the API Connect user interfaces fails when using the Safari web browser

If you are using the Safari web browser and a Basic Authorization header exists for the same DNS domain in which API Connect is running, attempts to log in to the API Connect user interfaces, or to sign up by using an activation link, fail. To avoid this problem, use an alternative web browser.

Endpoints cannot be changed by using APICUP after they have been configured in Cloud Manager

The changing of endpoints with APICUP after they have been configured in Cloud Manager is not supported. Any such endpoint changes will not be propagated to a running deployment.

An OAuth provider fails if the resources that it references aren't enabled in the Catalog

If you enable an OAuth provider in a Catalog then any resources that it references, such as API user registries or TLS client profiles, must be enabled in the same Catalog; if not, then although the OAuth provider might publish successfully it will fail at run time. For information on enabling resources in a Catalog, see [Creating and configuring Catalogs](#).

Redact conditions within `switch`, `operation-switch`, or `if` policies might not execute after converting to DataPower API Gateway

If an API Connect v5-compatible `redact` policy is found within a `switch`, `operation-switch`, or `if` policy, the migration utility does not move the `redact` policy to the beginning or end of the assembly. The difference in the response between API Connect v5 and DataPower API Gateway may prevent data from being redacted in the DataPower API Gateway.

For example, if an assembly includes a `switch` policy containing four redact conditions followed by an `invoke` policy, each redact condition redacts the response data. After porting to the API Gateway, the redact conditions remain inside the `switch` policy and target the `message.body` property for redaction. These redactions fail to execute because the `message.body` property has not yet been retrieved by the `invoke` policy. To correct this problem, you must move the `invoke` policy before the `switch` policy in the assembly.

Redact on invalid XPath fails after converting to DataPower API Gateway

A `redact` policy containing an XPath with a trailing slash is converted by the migration utility `apicm archive:port-to-apigw` command without removing the slash. This path is invalid. When the API containing this policy is called, an `assembly redact` error is generated.

"Test" tab's Trace feature displays blank policies list in Firefox

When you create an API, you can invoke it and debug its execution using the Test tab. The Test tab shows the API's response and includes a subtab where you can run a trace on the API's translation flow. In Firefox, the Trace tab's list of policies is empty. Use Chrome to debug your API so you can see the policies list in the trace.

Analytics REST API support for multiple query parameters

It is not possible to use multiple query parameters for the same nested API event record field when making calls to the analytics REST API. For example:

```
/cloud/events?response_http_headers[X-Client-IP]=not:10.51.11.183&response_http_headers[X-Client-IP]=not:10.51.11.184
```

Not possible to switch analytics from a three replica deployment to a one replica deployment

An error is logged in the storage pods when this is attempted. For a workaround, see: <https://www.ibm.com/support/pages/node/6845818>.

Limitations to support for JSONata

The DataPower API Gateway support for JSONata notation has the following limitations and restrictions.

- You cannot extract or redact content from the same source that is defined in the root. For example, if the root is defined as `message.body.a`, the action fails if the content to extract or redact is defined as the entire input.

As a workaround, remove the last element of the root and define that element as the content to extract or redact. For the previous example, define root as `message.body` and the content as `a`.

- New JSON objects cannot be constructed from the capture field that is specified in an Extract policy. As a workaround, use the transform expression `$merge ([])` to construct the empty object. You can then specify more extracts as needed for key-value pairs.
- If soft linked copies of a field exist in different locations, an Extract or Redaction policy can cause content from all of the linked fields to be extracted or redacted. As a workaround for the Extract policy, use the `$merge ($spread ($))` object function to construct new, unlinked objects and arrays. For example, the following transform expressions construct unlinked copies of the field `e`.

```
\ "a\" : $merge ($spread ($ . c))
\ "b\" : $merge ($spread ($ . c))
```

API Connect concepts

To help you get started, read about the API Connect concepts and obtain a high level understanding of the API management solution.

- **[Who does what in API Connect?](#)**
Review the different functions that users can perform in API Connect, the typical set of tasks that is performed by each type of user, and the documentation most often used for each task.
- **[Packaging strategy and terminology in API Connect](#)**
API Connect uses a proprietary packaging strategy for creating and publishing collections of APIs.
- **[API Connect components](#)**
The API Connect components provide a unified user experience across the API lifecycle. Changes in one stage of the API lifecycle are automatically reflected in the other components of API Connect.
- **[API Connect support](#)**
If you experience a problem with IBM API Connect that you cannot resolve, you can check the IBM Support website for the most recent technical bulletins and fixes. Otherwise, you can contact IBM Support.

Who does what in API Connect?

Review the different functions that users can perform in API Connect, the typical set of tasks that is performed by each type of user, and the documentation most often used for each task.

Systems administrators

If you deploy API Connect on premises, then your company manages the systems. Because API Connect comprises multiple servers even in an on-premises deployment, the collection of servers and services is referred to as a (local) cloud. This is unrelated to any servers and systems that are deployed in IBM® Cloud.

In an on-premises deployment, there are two roles for managing your deployment:

- **Cloud owner:** There is one cloud owner for a deployment, and that person adds other people to the server management team.
- **Cloud administrator:** Your deployment needs at least one cloud administrator; your company will probably assign multiple people to perform this role. Cloud administrators manage the deployment and configuration of API Connect servers and software.

In the API Connect documentation, cloud owners and administrators are primarily interested in the following sections, which introduce [concepts](#) and explain how to [deploy, maintain](#), and [configure](#) the components of an on-premises deployment.

Users

At the highest level, API Connect users belong to organizations, which are groups of people. A *provider organization* (often shortened as *p-org*) is a group of people who create, publish, and maintain APIs that are then used by people in a *consumer organization*. Consumers develop their own applications that call the APIs created by providers. A single customer can create multiple provider organizations and consumer organizations. For example, different divisions within a large company might work with different technologies and create entirely different sets of APIs, so it makes sense to group them into different p-orgs. Each provider organization publishes their APIs to a Developer Portal that uses consumer organizations to manage the customers who use those APIs.

At a minimum, you need one provider organization if you want to publish and manage APIs with API Connect, and one consumer organization for each customer.

Provider organization

A provider organization performs many tasks during the lifecycle of an API, from developing, publishing, and maintaining APIs to managing the membership of the organization itself. Each p-org can be as large or as small as needed. A large company might create a p-org for each product team, or for each department but a small company might use a single p-org.

A person can be a member of multiple provider organizations and receive a different set of permissions for each. When the user logs in to API Connect, they select the organization whose resources they will use, and receive the appropriate permissions for that organization for the duration of the session.

The following roles are commonly associated with a provider organization but are not necessarily represented by different people. In a small company, one person might perform all of the API maintenance tasks while in a larger company, several people might take on separate roles.

- **Organization Owner:** Every provider organization requires one owner to ensure that a user account is associated with the organization. The organization owner has full access to all of the p-org's resources (APIs, Products, Catalogs, and so on) and role cannot be deleted. The organization owner can add people to the organization manager role.
- **Organization Manager:** Maintaining the organization account is the job of the organization manager. In a large company, several people might share this role. Organization managers add *members* to provider organizations and assign them the permissions needed to perform their jobs. Organization owners and managers typically use the docs on [administering provider orgs](#) and [administering members and roles](#).
- **API Developer:** In API Connect, an API developer creates new APIs and updates existing APIs as needed. The API developer can configure *policies* that define security restrictions, logging rules, and quotas that control access to your company's resources. The API developer might be assigned to work with a specific *Catalog* that manages a subset of APIs, or they might receive access to all of the p-org's Catalogs and the *Spaces* within each. API developers are usually most concerned with documentation related to [developing APIs](#), [managing API settings](#), and [authoring policies for APIs](#).
- **API Administrator:** After an API developer creates a set of APIs, the API administrator manages the distribution. The API administrator defines *Plans* that determine access to the APIs, collects related APIs into *Products*, and publishes them to the consumer Developer Portals. To complete these tasks, the API administrator uses documentation focused on [managing API syndication](#) and [working with Products](#).
- **Product Manager:** A product manager controls access to the Developer Portal where customers subscribe to APIs and then tracks API usage and performance. For each customer, the product manager creates a consumer organization, assigns a customer representative as the owner, and manages the relationship between the provider organization and each consumer organization. The product manager can customize the appearance of Developer Portal for their p-org as well as provide forums and blogs for their consumers. The product manager uses the API Connect Analytics service to track API usage and

performance so that the p-org knows when APIs should be updated or retired. The product manager typically uses documentation on [administering consumer organizations](#), [using the Developer Portal](#), and [reviewing API analytics](#).

Consumer organization

A consumer organization creates applications that use the APIs that are developed by provider organizations. Each provider's APIs are made available to each consumer organization in the API Connect Developer Portal. Members of a consumer organization typically perform two functions:

- Consumer Organization Owner: The consumer organization owner also manages the organization by inviting other members and assigning permissions to each. This role additionally has permissions to customize the Developer Portal where members access APIs that the organization is subscribed to.
- Application Developer: Create applications that invoke APIs shared through the Developer Portal.

The only API Connect feature that API consumers use is the Developer Portal, which is managed by the API provider. There is no need for consumers to read the API Connect documentation.

- [API Connect user roles](#)

The IBM API Connect solution provides an infrastructure, tools, and facilities that allows users to create, manage, and stage APIs. The ability to perform tasks in the API Connect user interfaces is controlled through user roles, and the permissions that are assigned to those roles.

API Connect user roles

The IBM® API Connect solution provides an infrastructure, tools, and facilities that allows users to create, manage, and stage APIs. The ability to perform tasks in the API Connect user interfaces is controlled through user roles, and the permissions that are assigned to those roles.

The roles described here are the default API Connect roles. In the API Manager user interface, you can create custom roles; for more information, see: [Creating custom roles](#). You can also create custom roles in the Developer Portal user interface.

The following sections describe the roles and permissions for each of the API Connect user interfaces:

- [User roles in the Cloud Manager UI](#).
- [User roles in the API Manager UI](#).
- [User roles in the Developer Portal UI](#).

User roles and permissions in the Cloud Manager UI

The following table describes the Cloud Manager UI user permissions as configured in the base product. Certain roles can be edited as indicated in Table 2, and custom roles can be created. For instructions on how to create custom roles for the Admin organization (Cloud Manager users), see [Creating roles in the admin organization](#).

Table 1. Cloud Manager UI permissions

Permission	Action	Meaning
Cloud Settings	View	View all items on the Cloud Manager > Settings menu (except Roles)
	Manage	Add, update, and delete all items on the Cloud Manager > Settings menu (except Roles)
Member	View	View members on the members list located at Cloud Manager > Members
	Manage	Add and invite members from Cloud Manager > Members Note: By default, a user with Member > Manage permission can assign, to themselves or to another user, any role with any permission regardless of the permissions that they themselves have. However, you can apply a restriction such that, for a user to assign a role, they must themselves have at least all of the permissions that are applied to that role. To apply that restriction, complete the following steps: <ol style="list-style-type: none"> 1. Log in to your management server from the toolkit CLI as a member of the cloud administration organization; for details, see Logging in to management server. 2. Enter the following command (the terminating hyphen character means that the command takes input from the command line): <pre>apic cloud-settings:update --server mgmt_endpoint_url -</pre> where <code>mgmt_endpoint_url</code> is the platform API endpoint URL. 3. Enter the following data, followed by a new line: <pre>restrict_member_manage_permission: true</pre> 4. Press CTRL D to terminate the input.
Org	View	Org:View is a permission assigned to all Roles in Cloud Manager. It does not provide access to any functionality. It allows a user to activate their membership. It is the only permission in the Member role.
Provider-Org	View	View the list of provider organizations at Cloud Manager > Provider Organizations
	Manage	Add, edit, and delete provider organizations and invite owners from Cloud Manager > Provider Organizations
Settings	View	View the items on the Cloud Manager > Resources menu plus Roles located at Cloud Manager > Settings > Roles
	Manage	Add, edit, and delete the items on the Cloud Manager > Resources menu plus Roles located at Cloud Manager > Settings > Roles
Topology	View	View the items on the Cloud Manager > Topology menu
	Manage	Add, edit, and delete the items on the Cloud Manager > Topology menu
Analytics	View	View items on Cloud Manager > Analytics, and also create, update, duplicate, delete, share, and unshare saved queries.

The following table lists the various Cloud Manager UI roles and the permissions assigned to them.

Table 2. Cloud Manager UI roles

Role	Permissions	Actions	Default role provides access to	Notes
Owner	All permissions	All actions	All menus	Cannot be modified or deleted.

Role	Permissions	Actions	Default role provides access to	Notes
Administrator	All permissions	All actions	All menus	Includes analytics:view on new v10.0.5 installations, this permission must be manually added on installations upgraded from earlier releases.
Member	Org	View	Membership activation only	Cannot be modified or deleted. Member role is automatically assigned to all users when they activate their membership from the invitation. It allows them to activate but does not provide access to any menus.
Organization Manager	Org	View	N/A	Can be modified and deleted.
	Provider-Org	View, Manage	Provider Organizations menu	
Topology Administrator	Org	View	N/A	Can be modified and deleted.
	Topology	View, Manage	Topology Menu	
	Settings	View, Manage	Resources menu plus Settings > Roles	
Viewer	All permissions	View	All menus, view only	Cannot be modified or deleted.

User roles and permissions in the API Manager UI

The following tables describe the API Manager UI user permissions.

A user with Roles permission can change the permission assignments, and can create custom roles; for more information, see [Creating custom roles](#) in the section, [Managing your APIs](#).

Note: In API Manager, the Owner role has full access and cannot be edited or deleted. All other roles, including custom roles, can be deleted. If you delete a role, users lose that role. If a user loses that role, their account remains in API Manager, enabling you to add a role to the user at a future date.

Table 3. Organization permissions

Permissions	Action	Permits the member to
Member	View	View organization's members
	Manage	<p>Manage organization's members</p> <p>Note: By default, a user with Member > Manage permission can assign, to themselves or to another user, any role with any permission regardless of the permissions that they themselves have. However, you can apply a restriction such that, for a user to assign a role, they must themselves have at least all of the permissions that are applied to that role. To apply that restriction, complete the following steps:</p> <ol style="list-style-type: none"> 1. Log in to your management server from the toolkit CLI as a member of the cloud administration organization; for details, see Logging in to management server. 2. Enter the following command (the terminating hyphen character means that the command takes input from the command line): <pre>apic cloud-settings:update --server mgmt_endpoint_url -</pre> <p>where <i>mgmt_endpoint_url</i> is the platform API endpoint URL.</p> 3. Enter the following data, followed by a new line: <pre>restrict_member_manage_permission: true</pre> 4. Press CTRL D to terminate the input.
Settings	View	View an organization's configuration settings, including roles, TLS profiles, and user registries. View configuration settings for a Catalog or Space, including policies and OpenAPI extensions.
	Manage	Manage an organization's configuration settings, including roles, TLS profiles, and user registries. Manage configuration settings for a Catalog or Space, including policies and OpenAPI extensions.
Topology	View	Same permissions as Settings: View .
	Manage	Same permissions as Settings: Manage .
Org	View	View an organization
Product-Drafts	View	View draft APIs and Products
	Edit	View draft APIs and edit draft Products
Api-Drafts	View	View draft APIs
	Edit	Edit draft APIs and view draft Products
Product	View	View Products
	Stage	Stage Product
	Manage	Manage Product
Product-Approval	View	View Product lifecycle changes
	Stage	Approve the staging of a Product
	Publish	Approve the publishing of a Product
	Supersede	Approve the superseding of a Product
	Replace	Approve the replacement a Product
	Deprecate	Approve the deprecation of a Product
	Retire	Approve the retiring of a Product

Permissions	Action	Permits the member to
Consumer-Org	View	View consumer organization and developers
	Manage	Manage consumer organization and developers
App	View	View both production and development applications.
	Manage	Manage both production and development applications. A member with this permission can also request the promotion of a development app to a production app. This request triggers a task that needs approval by a member with the App-approval Manage permission.
App-Dev	Manage	Same permissions as Settings: Manage .
App-Approval	View	View application approvals, for requests to promote a development app to a production app.
	Manage	Manage (Approve or Decline) requests for approval to promote a development app to a production app.
Subscription	View	View application Plan subscriptions that have been created by application developers in the Developer Portal.
	Manage	Manage the application Plan subscriptions that have been created by application developers in the Developer Portal. The Manage permission includes ability to migrate a subscription to another plan.
Subscription-Approval	View	View application Plan subscription approvals.
	Manage	Manage (approve or decline) application Plan subscriptions.
Consumer-Onboard-Approval	View	View consumer onboard approvals.
	Manage	Manage (approve or decline) consumer onboard approvals.
Api-Analytics	View	View analytics data, as well as access and apply saved analytics queries.
	Manage	In addition to the view permissions, the user can create, update, duplicate, delete, share, and unshare saved analytics queries.
Child	View	At the provider organization level, view Catalogs in the provider organization. At the Catalog level, view Spaces in the Catalog.
	Create	At the provider organization level, create Catalogs in the provider organization. At the Catalog level, create Spaces in the Catalog.
	Manage	At the provider organization level, manage Catalogs in the provider organization. At the catalog level, manage Spaces in the Catalog. Management tasks including deleting a Catalog or Space, or transferring ownership of a Catalog or Space.

A user with Settings > Manage permission can change the permission assignments, and can create custom roles; for more information, see [Creating custom roles](#) in the section, [Managing your APIs](#).

Table 4. Default API Manager UI roles and the default permissions assigned to those roles.

Role	Role description	Permissions	Actions
Organization Owner	A provider organization owner has the full set of access permissions to API Connect functions, and also commission APIs and tracks their business adoption.	All permissions	All actions.
Administrator	A provider organization administrator has, by default, the full set of access permissions to API Connect functions, and also commission APIs and tracks their business adoption.	All permissions	All actions.
API Administrator	API administrators manage the lifecycle of APIs and publish APIs for discovery and use.	All permissions	All actions except cannot manage the following permissions: Member, Settings, Topology, and Child.
Community Manager	A community manager manages the relationship between the provider organization and application developers, provides information about API usage, and provides support to application developers.	Member	View
		Settings	View
		Topology	View gateway services or portal services at the provider organization.
		Org	View
		Drafts	View, Edit
		Product	View
		Product-approval	View
		Consumer-org	View, Manage
		App	View, Manage
		App-dev	Manage
		App-approval	View, Manage
		Subscription	View, Manage
		Subscription-approval	View, Manage
		Consumer-onboard-approval	View, Manage
		Api-analytics	View, Manage
		Child	View

Role	Role description	Permissions	Actions
Developer	API developers design and develop APIs and applications for the provider organizations to which they belong. Note: The Developer role allows the creation of Products and APIs, and the staging and publishing of Products to a Catalog or Space, when assigned to a user at the provider organization level--but not when assigned to a user who is a member only of a Catalog or Space within a provider organization. A Developer in a Catalog or Space can manage Products that are staged or published to the Catalog or Space.	Member	View
		Settings	View
		Topology	View gateway services or portal services at the provider organization.
		Org	View
		Drafts	View, Edit
		Product	View, Stage, Manage
		Product-approval	View, Stage, Publish, Supersede, Replace, Deprecate, Retire
		Consumer-org	View
		App	View, Manage
		App-dev	Manage
		App-approval	View, Manage
		Subscription	View, Manage
		Subscription-approval	View, Manage
		Api-analytics	View, Manage
		Child	View, Create
Member	Member of a provider organization	Org	View
Viewer	Viewer of a provider organization	Member	View
		Topology	View gateway services or portal services at the provider organization.
		Org	View
		Drafts	View
		Product-approval	View
		Consumer-org	View
		App	View
		App-approval	View
		Subscription	View
		Subscription-approval	View
		Api-analytics	View
		Child	View

Note: In API Manager, the Organization Owner role has full access and cannot be edited or deleted. All other roles, including custom roles, can be deleted. If you delete a role, users lose that role. If a user loses that role, their account remains in API Manager, enabling you to add a role to the user at a future date.

User roles in the Developer Portal UI

The following table describes the various Developer Portal UI roles that relate to working with APIs and applications. In addition, you can create custom roles for the Developer Portal site itself.

Table 5. Developer Portal UI roles

Role	Role Description	Permission	Actions
Owner	Owns and administers the app developer organization	Organization member	View, Manage
		Organization settings	View, Manage
		Organization view	View
		Consumer product	View
		Consumer app	View or Manage production or development applications
		Consumer app-dev	Manage development applications

Role	Role Description	Permission	Actions
		Consumer subscription	View or Manage the application Plan subscriptions that have been created by application developers in the Developer Portal. The Manage permission includes ability to migrate a subscription to another plan.
		Consumer app-analytics	View application analytics
Administrator	Administers the app developer organization	Organization member	View, Manage
		Organization settings	View, Manage
		Organization	View
		Consumer product	View
		Consumer app	View, Manage production or development applications
		Consumer app-dev	Manage development applications
		Consumer subscription	View or Manage the application Plan subscriptions that have been created by application developers in the Developer Portal. The Manage permission includes ability to migrate a subscription to another plan.
		Consumer app-analytics	View application analytics
Developer	Builds and manages apps in the developer organization	Organization member	View
		Organization settings	View
		Organization	View
		Consumer product	View
		Consumer app	View, Manage production or development applications
		Consumer app-dev	Manage development applications
		Consumer subscription	View or Manage the application Plan subscriptions that have been created by application developers in the Developer Portal. The Manage permission includes ability to migrate a subscription to another plan.
		Consumer app-analytics	View
Member	Member of the app developer organization	Organization	View
Viewer	Viewer of the app developer organization	Organization member	View
		Organization settings	View
		Organization	View
		Consumer product	View
		Consumer app	View applications
		Consumer production-app	View production applications
		Consumer app-analytics	View application analytics

Note: A user called admin is created automatically, with full administrator access to the Developer Portal site. The admin user can view Products and APIs but has no access to use APIs. The admin user assumes the email address of the owner of the provider organization associated with the Developer Portal.

Packaging strategy and terminology in API Connect

API Connect uses a proprietary packaging strategy for creating and publishing collections of APIs.

The packaging strategy supports API providers in meeting the requirements of the API consumers. An understanding of the concepts and terminology behind the packaging strategy is required before developing and deploying APIs using IBM® API Connect.

The following sections describe the concepts and terminology behind the packaging strategy for IBM API Connect:

- [APIs](#)
- [Plans and Products](#)
- [Catalog and Spaces](#)
- [Organizations and users](#)
- [LoopBack® applications](#)
- [Sample provider organization with two Catalogs](#)

APIs

An Application Programming Interface (API) is an industry-standard software technology. An API is a set of routines, protocols, and tools for building software applications. An API specifies how software components interact and provides quick access to common assets and processes. APIs can be public (such as offered on GitHub), can require client credentials, or can be kept private within an application. Thus, APIs are classified as external (public), partner (protected), or internal (private), based on how they are consumed.

An API is composed of operations, called methods, which are offered in one of the following styles in API Connect:

- A REST API is structured according to the principles of Representational State Transfer. REST APIs use HTTP or HTTPS requests to PUT, GET, POST, and DELETE data (also referred to as CRUD operations). REST identifies resources using URIs. Data can be described in a variety of formats (XML, HTML, JSON, TXT, etc.), with JSON being the popular choice. REST APIs specify MIME (Multipurpose Internet Mail Extensions) types. REST is platform- and language-independent and works across firewalls using HTTPS. REST APIs leverage HTTP standards for security, caching, and status codes. HTTP clients and servers are available for all major programming languages and operating system/hardware platforms. REST implementations are easily scaled due to use of HTTP and browsers as a uniform interface.
- A SOAP (Simple Object Access Protocol) API is a web service that is exposed as an API. SOAP interfaces are described in WSDL (Web Services Description Language) format. The WSDL is an XML document describing the structure for headers, messages, URL endpoints, and datatypes used to access a web service. SOAP is considered more secure than REST as it supports WS-Security as well as SSL. SOAP also contains WS-Reliable Messaging for reliability, rather than relying on retrying the operation (as does REST). SOAP requires a client application and is better suited for enterprise applications that require secure transactions.

APIs can be versioned and packaged into multiple Products for distribution to API consumers on the Developer Portal. For information about creating and managing APIs, see [Developing your APIs and applications](#) and [Managing your APIs](#).

Plans and Products

Plans and Products are proprietary packaging constructs that are unique to API Connect. API providers use Products to offer one or more APIs to the application developers who will consume the APIs (API consumers). The providers use Plans to control access to APIs and to manage API usage. Products are packages that contain both the APIs and the accompanying Plans. See [Working with Products](#).

To make an API available to an application developer, it must be included in at least one Product and at least one Plan.

Plans perform the following functions:

- Control which APIs an application developer can use
- Make available a collection of operations from one or more APIs
- Apply rate limits to APIs to differentiate between offerings
- Implement different rate limits to specify how many requests a consuming application is allowed to make during a specified time interval

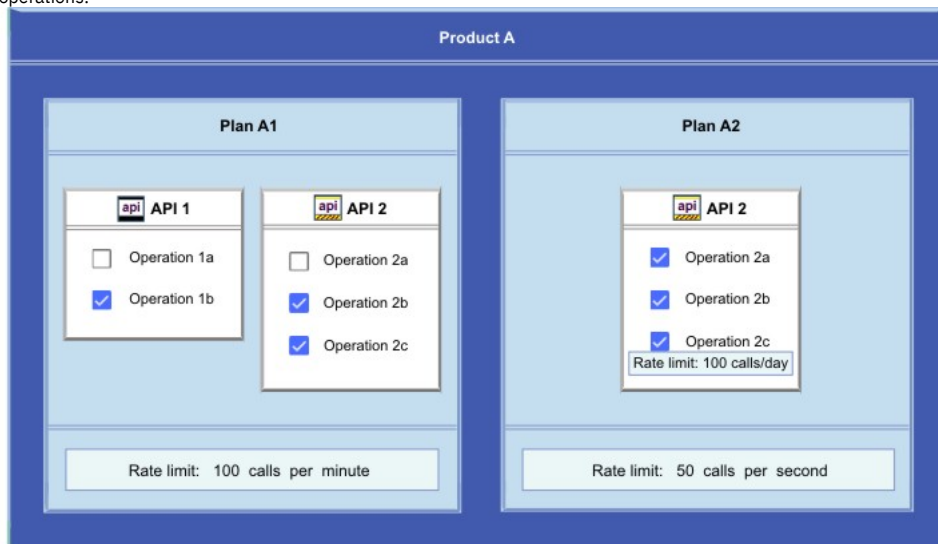
A rate limit can be implemented as a default rate that is shared across all operations in a Plan, or can be set for specific operations of an API. Plans can use differing rate limits to provide different levels of service to API consumers. For example, a "Demo Plan" might enforce a rate limit of ten calls per minute, while a "Full Plan" might permit up to 1000 calls per second.

A Product in API Connect bundles a set of APIs and Plans into one offering that is intended for a particular use. You can create Plans only within Products, and these Products are then published in a Catalog. The following rules apply for the relationship between Products and Plans:

- A Plan can belong to only one Product.
- A Product can have multiple Plans that each contain a different set of APIs.
- A Plan in one Product can share APIs with Plans from any other Product.

Multiple Plans within a single Product provide different levels of performance for the same offering. For example, a Product can include a "Demo Plan" that makes a single API available, and a "Full Plan" that makes several APIs available.

The following diagram illustrates how Plans can be used to make operations from one or more APIs available, and to set rate limits on both Plans and individual operations:

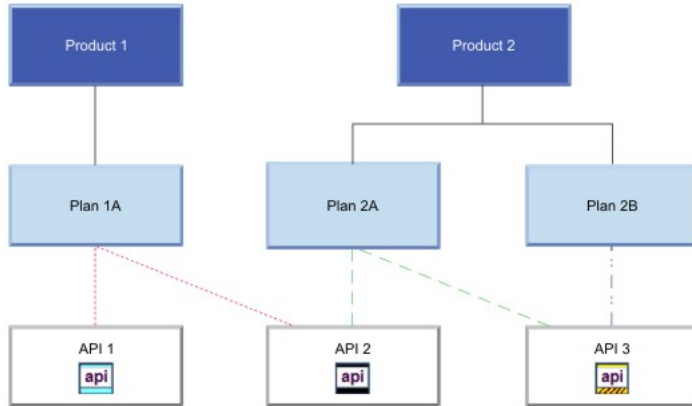


The diagram illustrates the following concepts:

- Product A contains two APIs and two Plans.
- Both APIs are included in Plan A1.
- Plan A2 includes only API 2, which is also included in Plan A1. Operation 2a for API 2 is excluded from Plan A1. All operations in API 2 are included in Plan A2.
- Different rate limits are set for the two Plans at the Plan level. For API 2 in Plan A2, a rate limit is also set specifically for Operation 2c to override the rate limit set at the Plan level.

Products are used to manage the lifecycle of the APIs they contain. The states in the Product lifecycle include draft, staged, published, deprecated, and retired. A Product in draft state is moved to the staged state when it is staged to a *Catalog*. A Product moves to the published state when the Product is published. The APIs in a Product become accessible to API consumers when the Product is in the published state and they then become visible on a *Developer Portal*. After a Product is published, application developers can use the Developer Portal to gain access to its APIs by registering applications to one or more Plans in the Product.

The following diagram illustrates the hierarchy of Products, Plans, and APIs.



For more information about managing the lifecycle of Products, see [Working with Products in the API Manager](#).

Catalogs and Spaces

A Catalog contains a collection of Products. Catalogs are staging targets through which Products (together with the accompanying Plans and APIs) are published on a Developer Portal. Catalogs are used to separate Products and APIs for testing before publishing them on a Developer Portal. In a typical workflow, an API provider uses a development Catalog when developing and testing APIs, and uses a production Catalog for publishing APIs that are ready for external use. Each Catalog has an associated Developer Portal for exposing the published Products. A Catalog includes runtime capability through an associated gateway service that handles any API requests for the APIs in that Catalog.

API Connect includes a syndication feature that enables API providers to partition a Catalog into multiple staging targets (or *Spaces*) for API development purposes. Each API provider development team can use its own dedicated Space to manage its Products independently of other teams. A Space has its own set of capabilities relating specifically to the Products and APIs that are created and published to that Space. Products and APIs in all Spaces in a given Catalog are published to the same Developer Portal. Spaces are not visible on the Developer Portal. Application developers who consume the APIs on the Developer Portal are unaware of the Space configuration used by the API Developers. On the Developer Portal, the APIs are seen as a coordinated offering within a Catalog.

For more information about Catalogs and Spaces, see [Working with Catalogs](#) and [Using syndication to partition Catalogs into Spaces](#).

Organizations and users

In the context of API Connect, there are two types of organizations: provider and consumer. An organization can encompass a project team, department, or division.

A provider organization owns APIs, and associated Plans and Products, and can additionally own *provider* applications that are called by APIs. To complete the various functions in the API lifecycle, a provider organization assigns responsibilities for certain tasks. Some standard responsibilities within a provider organization include:

- A provider organization owner who has the full set of access permissions to API Connect functions, and can also commission APIs and track their business adoption.
- API developers who design and develop APIs and applications for the provider organizations to which they belong.
- An administrator who manages the lifecycles of APIs and publishes APIs for discovery and use.
- A "community" manager who manages the relationship between the provider organization and application developers, provides information about API usage, and provides support to application developers.

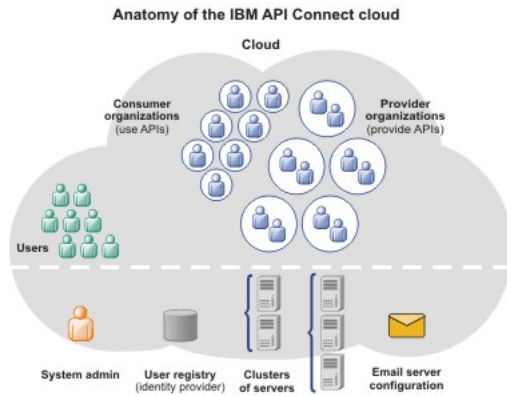
A consumer organization owns only *developer* applications, and consumes the APIs and applications produced by the provider organization. Standard responsibilities within a consumer organization include:

- A consumer organization owner who adds application developers to the consumer organization, views the Products and APIs that the provider organization has made available on the Developer Portal, and subscribes to APIs to use them in applications.
- Application developers who view the Products and APIs that the provider organization has made available on the Developer Portal, and subscribe to use these APIs in applications.

The provider and consumer organization responsibilities map to *roles* within API Connect. Some roles are independent of an organization; for example, an administrator who manages the cloud infrastructure and keeps the system running. For information about the full set of defined roles and access permissions, see [API Connect user roles](#).

Users have an existence in the API Connect ecosystem that is independent of an organization. A user can be a member of more than one provider or consumer organization.

There can be multiple provider organizations in one API Connect cloud, to provide an API development environment for each line of business of an enterprise. The API Connect cloud is a collection of servers that comprise an API Connect installation, including the configuration information and metadata that they contain. The cloud infrastructure is shared by all organizations, and managed independently of them (by a cloud or system administrator). The following diagram shows the relationship between the provider organizations, consumer organizations, and users. The clusters shown are logical groupings of servers with the same capability.



For more information about organizations, see [Administering provider organizations](#) and [Administering consumer organizations](#).

LoopBack applications

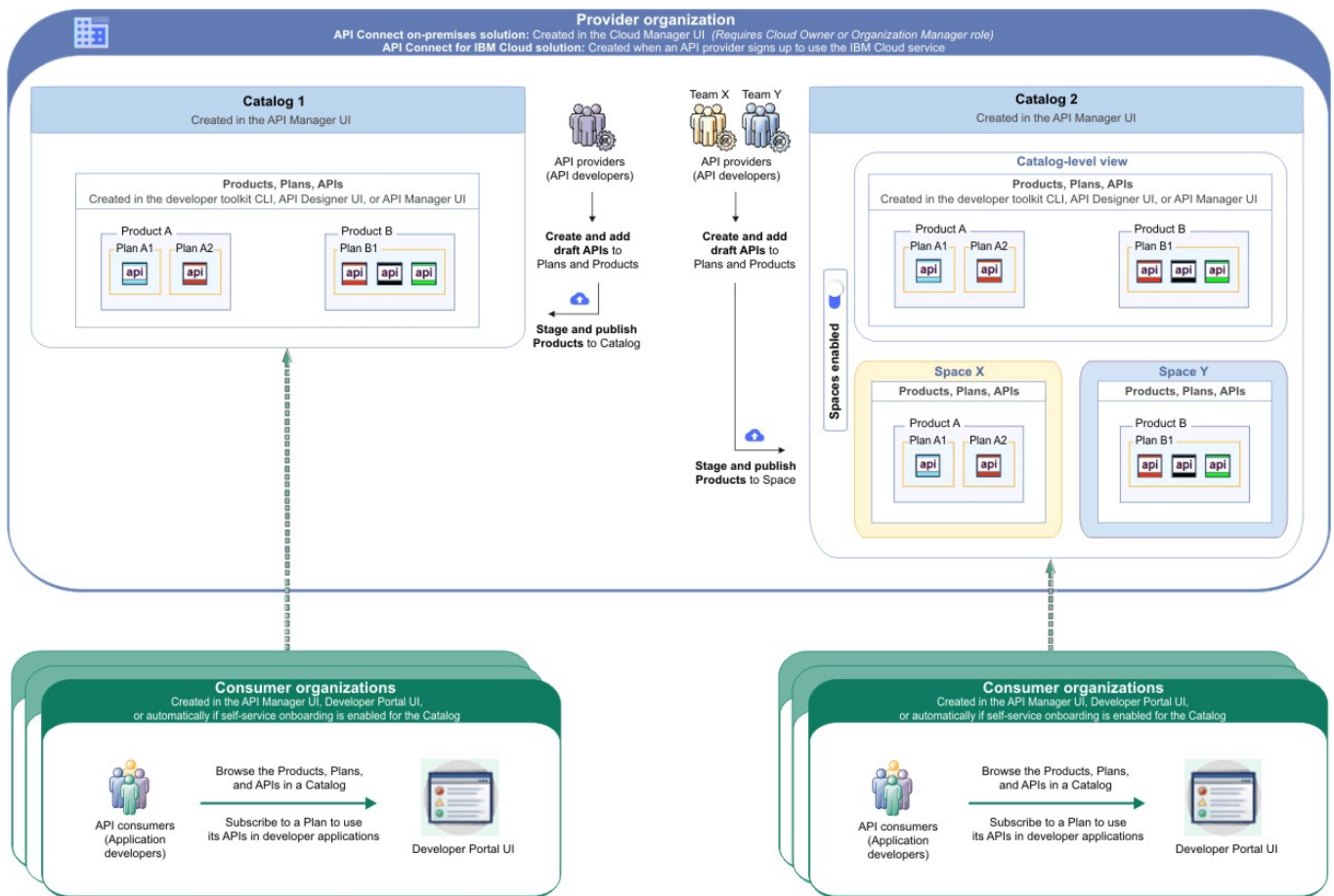
In addition to APIs, provider organizations can also create applications (with associated APIs), which are built using Node.js and Java technology. When published, these APIs and applications are called by developer applications. The developer applications contain client code that accesses APIs to interact with a service, system, or content. The developer applications are typically mobile or web applications that use the HTTP protocol.

For information about creating LoopBack applications, see [Creating APIs and applications](#).

Sample provider organization with two Catalogs

The following diagram shows an example for how APIs, Plans, Products, Catalogs, and Spaces fit within provider and consumer organizations. In this example, two Catalogs are created in the provider organization to act as staging targets for different sets of requirements.

- Catalog 1 does not require separation of the Products for use by individual provider development teams, and so does not have Spaces enabled. API developers with access to this Catalog create draft APIs, and stage and publish them as Products to the Catalog. Several consumer organizations are granted permission to explore, discover, and use the published Products and APIs. In each of the consumer organizations, the published Products and APIs from Catalog 1 are exposed on a single Developer Portal.
- Catalog 2 is partitioned into two Spaces (X and Y) so that two provider development teams can manage their Products independently. These teams stage and publish their APIs (as Products) separately to the individual Spaces, to make them accessible to application developers in multiple consumer organizations. In each of the developer organizations, the published Products and APIs from both Spaces are exposed in the same Developer Portal, and application developers who access this portal will see the APIs from both Spaces as a coordinated offering.



- [Understanding rate limits for APIs and Plans](#)

In API Connect, you can configure rate limiting on APIs and Plans to manage network traffic and API usage.

Understanding rate limits for APIs and Plans

In API Connect, you can configure rate limiting on APIs and Plans to manage network traffic and API usage.

What are rate limits and burst limits?

A rate limit is the maximum number of calls you want to allow in a particular time interval. Setting rate limits enables you to manage the network traffic for your APIs and for specific operations within your APIs. API Connect supports two types of rate limiting:

Rate limit

A specified number of calls to be accepted within a defined time period; for example, 100 calls per minute. In API Connect, rate limits can be defined as unlimited, or with a specified number of calls per second, minute, hour, day, or week.

Rate limits can be "hard" (enforced) or "soft". If the rate limit is hard and a call exceeds the limit, then the call is aborted and an error is returned. A soft rate limit allows the call to complete but logs a warning message. When a hard rate limit is reached, no more calls are accepted from that customer until the beginning of the next time period. For example, you might want to permit a total of 1000 calls per hour (rate limit). If a customer makes 1000 calls in the first 10 minutes, they cannot complete any more calls until the hour has expired.

Burst limit

A rate limit that is applied to a very small time period. In API Connect, burst limits are defined for multiples of seconds or minutes. Burst limits are always hard. When a burst limit is reached, calls are not accepted until the time period has expired. Once that pause is over, calls continue to be accepted until a hard rate limit is reached. Configuring a burst limit prevents usage spikes and ensures that the rate limit is evenly spread across its overall time period. For example, you might want to permit a total of 1000 calls per hour (rate limit) and a maximum spike of 50 calls per second (burst limit).

Where can you apply rate limits?

You can configure rate limiting at several levels of an API and the Plan containing it. Each of the rate limits is tracked separately and when a hard limit is exceeded, calls at that point are no longer accepted until the start of the next time period.

The following list explains where you can configure rate limits:

Plan: all APIs

You can define a general rate limit at the Plan level, and it affects all of the consumer orgs that use that Plan. When a rate limit is enforced for a Plan, the limit is checked before every call that involves any API contained in that Plan. If the rate limit is exceeded, APIs are not even called. For information on setting a rate limit for a Plan, see [Editing a draft Product](#).

Note: If the subscriber changes to a different Plan, only the Plan-level rates limit are reset. If there is a hard rate limit specified in the API's definition or assembly, that rate limit is still enforced under the new Plan.

Plan: API path and operation

You can define a rate limit for a specific path and operation of an API within a Plan; for example, you might allow unlimited calls to a GET operation but enforce a limit on PUT operations for the same API. You can even define multiple rate limits on the same path and operation. If the rate limit is enforced, then the limit is checked before every call that involves the API's path and operation. For more information on configuring rate limits on API paths and operations within a plan, see the following topics:

- [Defining rate limits for an API operation](#)
- [Describing Plans in your Product](#)

API definition

You can define a rate limit on an API in the API definition (the Design page in the user interface). This rate limit applies to all calls involving the API, from all consumer orgs. If the containing Plan allows the call but the API definition has a hard limit set, the call is aborted. For more information on setting a rate limit on an API definition, see [Changing an API rate limit](#).

API assembly

When you apply a rate limit to an API assembly, the limit only affects calls at that point in the assembly. This allows you to define a more granular rate limit than if you set it on the API definition. If the Plan and the API definition allow a call to proceed but a hard limit that is set on the assembly is exceeded, the call is aborted at the point in the process flow where the rate limit is set. Any actions that appear in the assembly's process flow before the rate limit policy are executed. Only the actions that appear in the flow after the rate limit policy are aborted. For more information on configuring assembly-level rate limits, see the following topics:

- [Rate Limit policy](#)
- [GatewayScript policy](#)
- [Using context variables in a GatewayScript policy](#)

How can you test rate limits?

When you're ready to test rate limits on an API or Plan, make sure you deploy it in a test Catalog where automatic subscriptions are not enabled. Execute a larger number of calls than specified by your rate limit, within a shorter period of time. If you set soft rate limits, check for warning messages that indicate when each limit is exceeded. If you set hard limits, you can tell it was enforced when the API call fails. You can also check the error messages to verify which calls exceeded the limits.

The built-in test application that is used by the API Manager and API Designer Test tool is not subject to rate limits if you enabled automatic subscriptions for the Catalog where you are testing. To ensure that your rate limits are applied as intended, create a new test Catalog that requires manual subscriptions and test your API and Plan there. For more information on creating and configuring Catalogs, see [Working with Catalogs](#).

API Connect components

The API Connect components provide a unified user experience across the API lifecycle. Changes in one stage of the API lifecycle are automatically reflected in the other components of API Connect.

- [Cloud Manager](#)
- [The developer toolkit](#)
- [API Manager](#)
- [API Gateways](#)
- [Runtime](#)
- [Developer Portal](#)
- [API Analytics](#)
- [Typical tasks per interface component](#)
- [API Connect server requirements](#)

Cloud Manager

The API Connect Cloud Manager component is used to manage the API Connect on-premises cloud. The Cloud Administrator uses this UI to:

- Define the cluster of *Management servers*, *Gateway servers*, and *containers* that are required in the cloud, and configure the topology. For information about Management servers and Gateway servers, see [API Connect server requirements](#). For information about containers, see [Runtime](#).
- Manage (modify, move, remove, restart, reboot) the servers in the cloud.
- Monitor the health of the cloud.
- Define and manage the provider organizations that develop APIs. (Assigned managers or owners of provider organizations can also complete this task.)
- Define additional cloud administrators, or set up users with roles that enable access to specific capabilities.
- Add user registries for authenticating users and securing APIs, and configure the secure transmission of data (for example, through websites).

For more information about the Cloud Manager, see [Managing your cloud](#).

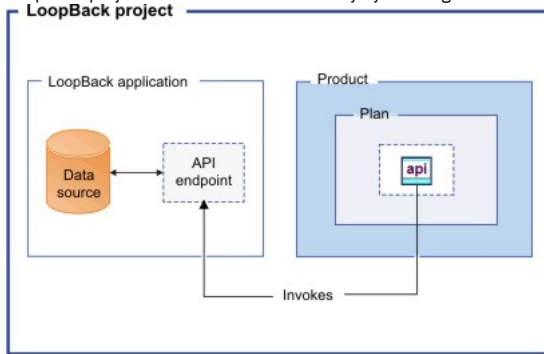
The developer toolkit

The developer toolkit provides the tools for modeling, developing, and testing APIs and LoopBack® applications. The developer toolkit includes a command-line interface (CLI). It also incorporates LoopBack, an open source Node.js framework.

API developers use the API management functions in the API Designer or the CLI to create draft API definitions for REST and SOAP APIs, or for OAuth provider endpoints that are used for OAuth 2.0 authentication. The API definitions can be configured to add the API to a Product, add a policy assembly flow (to manipulate requests/responses), and to define security options and other settings. APIs can then be tested locally prior to publishing, to ensure they are defined and implemented correctly.

Using LoopBack, an API developer can create a Node.js application, connect to a data source such as a back-end database or a REST API to be consumed, and then expose the application as a REST API by creating a model definition. A LoopBack model defines the application data, validation rules, data access capabilities, and business logic for an API, and provides a REST API by default. This REST API can then be used by a REST API definition that was created using the API Designer or CLI and

exposed to your users. The API and its associated application, which are implemented as a LoopBack project, must both be published to enable the project to be run. LoopBack projects can also be tested locally by creating a local runtime environment. The following diagram illustrates the LoopBack project architecture:



Draft APIs (in their containing Products) that are created by using the API Designer, CLI, or LoopBack are published to Catalogs. Applications created using LoopBack are published to containers, from where they run when called. (For information about containers, see [Runtime](#).)

The developer toolkit is installed locally, for offline API and application development. For more information about the developer toolkit, see [Developing your APIs and applications](#). For more information about LoopBack, see [LoopBack: The Node.js API Framework](#).

API Manager

The API Manager provides a user interface that facilitates promotion and tracking of APIs that are packaged within Products and Plans. API providers can move the Products through their lifecycle, and manage the availability and visibility of APIs and Plans.

Catalogs and Spaces are created in the API Manager to act as staging targets through which APIs, Plans, and Products are published to consumer organizations. API providers can stage their Products to Catalogs or Spaces, and then publish them to make the APIs in those Products visible on a *Developer Portal* for external discovery.

To control access to the available API management functions, users in the provider organization can be set up in the API Manager UI with assigned roles and permissions. API providers can also use the UI to manage the consumer organizations that sign up to access their APIs and Plans. Developer *communities* can additionally be created as a way of grouping together a collection of consumer organizations to whom a particular set of Products and Plans can be made available.

The API Manager UI also includes functions to manage the security of the API environment, and provides access to analytics information about API invocation metrics within customizable dashboard views.

For more information about the API Manager, see [Managing your APIs](#).

API Gateways

Gateways enforce runtime policies to secure and control API traffic, provide the endpoints that expose APIs to the calling applications, and provide assembly functions that enable APIs to integrate with various endpoints. They also log and report all API interactions to the API Connect analytics engine, for real-time and historical analytics and reporting. API Connect supports the following types of API gateways:

- DataPower Gateway (v5-compatible)

The DataPower® Gateway is an enterprise API Gateway that is built for departments and cross-enterprise usage. In V10 API Connect, the DataPower Gateway is compatible with the V5 DataPower Gateway, so it is referred to in documentation as the "v5-compatible" (or "v5c") gateway. The v5-compatible gateway provides a comprehensive set of API policies for security, traffic management, mediation, acceleration, and non-HTTP protocol support.

- DataPower API Gateway

The DataPower API Gateway was designed with the same security focus as the DataPower Gateway (v5-compatible). Where DataPower Gateway (v5 compatible) is built for flexibility, DataPower API Gateway is built specifically for the API use case, with resulting performance benefits. DataPower API Gateway is optimized for the cloud. Use this gateway if you are running applications in a public or private cloud and want to expose them as APIs.

Your API Connect deployment can include a virtual DataPower Gateway. Support for a physical DataPower Gateway is also available, subject to certain conditions.

For more information about the gateways, and a comparison of their features, see [Gateway types](#).

Runtime

You can run applications and API implementations in API Connect in a containerized runtime.

Containerized runtime

A containerized runtime environment provides a lightweight deployment location for APIs and applications. A container wraps an application in a complete file system that includes everything it needs to run, such as code, runtime, system tools, and system libraries. You can use Docker Swarm or Kubernetes containers to run your APIs and applications being managed by API Connect.

Developer Portal

The Developer Portal provides a customizable self-service web-based portal to application developers to explore, discover, and subscribe to APIs.

When API providers publish APIs in the API Manager, those APIs are exposed in the Developer Portal for discovery and usage by application developers in consumer organizations. Application developers can access the Developer Portal UI to register their applications, discover APIs, use the required APIs in their applications (with access approval where necessary), and subsequently deploy those applications.

The Developer Portal provides additional features, such as forums, blogs, comments, and ratings, for socialization and collaboration. API consumers can also view analytics information about the APIs that are used by an application, or used within a consumer organization. For more information, see [Developer Portal: Socialize your APIs](#).

API Analytics

API Connect provides the capability to filter, sort, and aggregate your API event data. This data is then presented within correlated charts, tables, and maps, to help you manage service levels, set quotas, establish controls, set up security policies, manage communities, and analyze trends. API analytics is built on the OpenSearch open source real-time distributed search and analytics engine.

API Connect server requirements

From an on-premises cloud, you can create, promote, use, and track APIs. An on-premises cloud is composed of various appliances, where each appliance is a server of a specific type. The collection of servers defines your cloud and determines how to distribute the work of managing, analyzing, routing, and storing data.

Your on-premises cloud can be a combination of new and existing physical appliances and virtual appliances, or can be entirely composed of virtual appliances. The type and quantity of servers in an API Connect environment are determined by the individual needs of each enterprise, but the minimum requirement is one Management server, one Analytics server, one Gateway server, and one server to host the Developer Portal.

The API Connect on-premises cloud includes the following server types:

- **Management server.** Stores all of the cloud configuration, and controls communication between the other servers within API Connect. Manages the operations of the various servers in the API Connect cloud and provides the tools to interface with the various servers. The Cloud Manager and API Manager user interfaces run on the Management server.
- **Analytics server.** Provides analytic functions that collect and store information about APIs and API users.
- **Gateway server.** Processes and manages security protocols and stores relevant user and appliance authentication data. The Gateway server also provides assembly functions that enable APIs to integrate with various endpoints, such as databases or HTTP-based endpoints.
- **Developer Portal server.** Provides a customizable social developer portal with a full-featured content management system, and includes clustering capability. Enables API providers to build portals for their application developers, and provides the interface for application developers to discover APIs and subscribe to usage Plans contained in the published Products for use in their applications.

Note: All Management appliances in an API Connect cloud must run at the same firmware level as each other. Gateway appliances can run on different firmware levels to each other, but it is recommended that all of the Gateway appliances run on the same level as each other.

Typical tasks per interface component

API Connect offers both command line and graphical user interfaces. Provider and consumer organizations use different interfaces for completing typical tasks. Refer to the following table to locate the interface that corresponds to a specific task.

Table 1. API Connect Tasks per interface component

Organization Type	Interface Component	Tasks
API Provider	Command Line Interface (CLI)	Create APIs, Plans, and Products
	API Designer UI	Create APIs, Plans, and Products
	API Manager UI	Create Catalogs and Spaces; Create Consumer Organizations
	Cloud Manager UI	Create Provider Organizations
API Consumer (application developer)	Developer Portal	Access APIs to create and run applications; Create Consumer Organizations

If self-service onboarding is enabled for a Catalog, a consumer organization is automatically created when an application developer signs up or is invited by the API provider to a Developer Portal, and the application developer then becomes the owner of that consumer organization.

- [Gateway types](#)
IBM API Connect provides the different types of gateways for use with your deployment.

Related concepts

- [Packaging strategy and terminology in API Connect](#)

Gateway types

IBM® API Connect provides the different types of gateways for use with your deployment.

Event Gateway Service

Event Endpoint Management in Cloud Pak for Integration provides an Event Gateway Service to control access from clients that consume AsyncAPI events from a Kafka cluster. For more information about event endpoints and the Event Gateway Service, see the [Cloud Pak for Integration documentation](#).

DataPower API Gateway

The DataPower® API Gateway has been designed with APIs in mind, and with the same security focus as DataPower Gateway (v5 compatible). Where DataPower Gateway (v5 compatible) was built for flexibility, DataPower API Gateway is built specifically for the API use case, with resulting performance benefits.

DataPower API Gateway was built and optimized for the cloud. Use this gateway if you are running applications in a public or private cloud and want to expose them as APIs.

DataPower Gateway (v5 compatible)

DataPower Gateway (v5 compatible) provides compatibility with the IBM DataPower Gateway that was provided with IBM API Connect Version 5 and earlier releases.

Consider using DataPower Gateway (v5 compatible) if you are an existing DataPower user and want to utilize your DataPower resources and knowledge.

Gateway comparison

The following table compares support for features between the gateway types. For v5-compatible features and policies that are not supported in DataPower API Gateway, you can use the API Connect Migration Utility (AMU) to migrate them to an API Gateway-compatible format. For more information, see [Migrating a Version 5 deployment](#).

Table 1. Comparison of the DataPower Gateway (v5 compatible) and DataPower API Gateway

Feature	DataPower Gateway (v5 compatible)	DataPower API Gateway
Native policies	No	Yes
OAuth provider	Full OAuth 2.0 Support	Full OAuth 2.0 Support
OAuth policy	No	Yes
OpenID Connect	Supported through a template	Supported natively
Invoke policy	Yes	Yes
Custom policies	Yes	Yes
Conditional policies	if, operation-switch, switch	switch
Activity logging	Implicitly executed at the end of API assembly	Configured in the API design, outside of the API assembly.
Parse policy (threat detection)	No	Yes
Gateway extensions	Yes	Yes
Support for mutual TLS (mTLS)	Yes	Yes

Attention: Version 5-compatible gateways and DataPower API Gateways have different mechanisms for interacting with the API context for GatewayScript and XSLT policies, or if you are authoring user-defined policies. Ensure that you are using the correct mechanisms for your gateway type:

- For v5-compatible gateways and v5 compatibility for APIs created for the API Gateway, see [GatewayScript code examples](#) and [XSLT policy examples](#).
- For DataPower API Gateway, see [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway](#).

API Connect support

If you experience a problem with IBM® API Connect that you cannot resolve, you can check the IBM Support website for the most recent technical bulletins and fixes. Otherwise, you can contact IBM Support.

For the most recent API Connect technical information, see the [IBM Support Community](#).

If you cannot find a resolution, before you contact IBM Support, you must gather information about the problem. For details of the information you must gather before contacting IBM Support, see [MustGather - Collecting data: API Connect problem determination](#).

Providing IBM Support with a good description of the problem together with the details of your API Connect configuration, helps to expedite the problem resolution.

API Connect glossary

The IBM® API Connect and Cloud Manager glossary of terms and definitions.

API Administrator (role)

Manages the API product lifecycle for the Provider Organizations for which they are a member.

API Event

An event captured for use by API Connect Analytics or third-party analytics, such as response time, HTTP response code, payload of the request and response body, and so on. An API event is logged each time an API operation is invoked via the Gateway server.

API Gateway

Service that acts as a single entry point or “front door” for provider APIs and back-end services. An API Gateway accepts and processes concurrent API calls, and performs traffic management, authorization and access control, monitoring, and API version management. See also [IBM DataPower Gateway](#).

API Manager

Graphical tool in API Connect that enables you to manage Catalogs, Spaces, and APIs, users and roles in the Provider organization, consumer organizations and communities, publish Products to the Developer Portal, and analyze API usage.

API Designer

Graphical tool that runs locally on a laptop or desktop system that enables you to create and modify APIs and LoopBack® apps, and publish them to IBM Cloud or container runtimes.

API operation

REST API call consisting of an HTTP verb and a URL path (endpoint). For example, `GET http://myserver.com/api/users` that returns a list of users.

Application

Software that consumes (calls) an API. One or more Applications are registered by a consumer organization to subscribe to APIs. The application is allocated client ID and client secret credentials that it supplies when invoking API calls.

assembly

Makes side calls to external services and then transforms and aggregates the response before a response is relayed to the calling application.

Availability zones

In API Connect these are logical groupings of API Connect services that will typically reflect the deployment environment. For example, you could have availability zones named **internal** and **external** to reflect the APIs published in each. Or they could be defined by the geographic location the services are running in.

Catalog

A staging target that behaves as a logical partition of the gateway and the Developer Portal. The URLs for API calls and for the Developer Portal are specific to a particular Catalog.

client ID

A piece of information that identifies an individual application. An application can invoke an API only if it passes an application key that is recognized by the IBM API Connect system and is granted access to the API. The application key is passed by the client by using an HTTP query parameter.

client secret

A piece of information used together with the application key to verify the identity of an application. An API can be configured to require that client applications supply their client secret with their client ID. The client secret functions effectively as a password known only to the application. The client secret is passed by the client by using an HTTP query parameter.

Cloud Manager (tool)

Graphical tool in API Connect on-premises installation that enables you to define servers, administer and scale system resources, monitor runtime health and create Provider organizations. The Cloud Administrator is the primary user of the Cloud Manager.

Cloud Administrator (role)

Manages the configuration of resources, regions, and availability zones for the Admin Organization of an on-premises installation.

cluster

A collection of one or more servers that provide a specific function.

Cold standby

A deployment configuration in which failover servers are in a stopped state until a failover is required, resulting in a longer time window to restore the normal operation of the solution.

community

A collection of consumer organizations. It is used as a grouping construct when publishing APIs. Communities are used to restrict the visibility and accessibility of APIs. An API can be published to selected communities, which means that only application developers within those organizations can see the API.

Community Manager (role)

Manages application developer communities for the Provider Organizations for which they are a member.

Consumer organization

Within a catalog, representation of a business entity that wishes to consume APIs exposed by the Catalog. For example, a third-party application development company would register themselves as a consumer organization (via the Developer Portal). Each developer in their company can then be registered as a user inside that consumer organization.

IBM DataPower® Gateway

API Gateway service component of API Connect that helps provide security, control, integration and optimized access to APIs. Due to its security and hardening, it is well-suited for a deployment in the demilitarized zone (DMZ) for externally-facing production scenarios. Available in physical, virtual, cloud, Linux and Docker form factors. There are two gateway types, DataPower Gateway (v5 compatible) and DataPower API Gateway; for details, see [Gateway types](#).

Developer (role)

Creates and configures APIs, Products, and policies for the Provider Organizations for which they are a member. An API Developer can be a member of one or more Provider Organizations. The API Developer focuses on the technical implementation of APIs more than they do on the business relationship with application developers.

Development Catalog

Catalog used for testing APIs that are under development and in which approvals are bypassed for publishing and lifecycle actions. Pending approvals are canceled when a non-Development Catalog is converted to a Development Catalog.

Developer Portal

Component of API Connect that provides a customizable graphical web portal for developers to discover APIs, register applications that consume APIs, subscribe to usage plans, and test and use APIs.

Developer Toolkit

Locally-installed package that includes API Designer and the `apic` command-line tool that enables you to create, edit, manage, and publish APIs and apps.

Gateway

See API Gateway.

Gateway service

API Connect service that provides API gateway functionality, such as microgateway or IBM DataPower.

Hot standby

A deployment configuration in which a set of servers are actively running ready to instantaneously take over in the event of a failure, but not actively serving traffic until that failover.

Identity provider

Provides identifiers for users looking to interact with a system, assert to such a system that such an identifier presented by a user is known to the provider, and possibly provide other information about the user that is known to the provider. API Connect supports LDAP, AuthURL, and Local User Registries.

Integrations

Third-party tools to build on and improve your API Connect workflow; for example, Slack, Auth0, etc.

Integration profile

Standard API that you can use to integrate with management services for things like identity, notifications, API analytics, and so on.

Keystore

A repository of security certificates, either authorization certificates or public key certificates, and corresponding private keys, used in SSL encryption. The Keystore file has a `.jks` extension.

LoopBack model

A JavaScript object that represents application data and includes validation rules, data access capabilities, and business logic. LoopBack models provide a REST API by default, and connect to data sources for access to back-end data

LoopBack data source

A JavaScript object that represents a back-end service such as a database, REST API (to be consumed), or SOAP web service. Data sources are backed by connectors that communicate directly with the database or other back-end service.

Management server

Stores all of the cloud configuration, and controls communication between the other servers within API Connect.

Management service

Consists of one or more Management servers.

Member (role)

The default role for all users in both the Admin and Provider Organization. All users are assigned the Member role in addition to other roles they may require.

microservice

Modular, independently-deployable application service that communicates with other microservices through a REST API. Microservices are typically organized around capabilities, for example, recommendation, inventory, shipping, or billing.

Mutual authentication

Process in which both entities on a network authenticate each other. In a network environment, the client authenticates the server and vice-versa. It is optional for TLS. Also called two-way authentication.

Notification settings

How you configure notifications for API Connect users (API providers and consumers).

Notification services

Integrations that provide notification capabilities, such as email.

Notification templates

The configuration of the message format and wording for a notification service.

OAuth provider

The OAuth provider supplies the OAuth authentication for logins. API Connect supports both Native and Third Party OAuth providers. Some common third-party OAuth providers are Google and Facebook.

OpenAPI Components

Part of API specification that contains a set of reusable objects for aspects of an API specification, such as schemas, responses, requestBodies, and headers. For more information, see [OpenAPI v3 specification](<https://github.com/OAI/OpenAPI-Specification/blob/OpenAPI.next/versions/3.0.md#components-object>).

organization

The entity that owns APIs or applications that use APIs. A provider organization owns APIs and associated Plans, and can additionally own applications. A consumer organization owns only applications. An organization has at least one owner. An organization can be a project team, department, or division.

Organization Manager (role)

The Organization Manager manages Provider Organizations for the Admin Organization.

Path

Defines the route through which users access REST APIs. A path consists of one or more HTTP operations such as GET or POST.

Plan

The packaging construct by which APIs are made available to developers. A Plan makes a collection of operations from one or more APIs available, and is published to communities of application developers. Application developers gain access to APIs by registering applications to access Plans. A Plan carries with it a collection of policy settings. In the simplest form, a Plan defines a single quota policy that applies to all the API operations that are accessed through the Plan. In more advanced cases, additional policies can be associated with a Plan.

policy

A configuration that controls a specific aspect of processing in the Gateway server during the handling of an API invocation at run time. Policies are the building blocks of assembly flows. Policies provide the means to configure capability, such as security, logging, routing of requests to target services, and transformation of data from one format to another. Policies can be configured in the context of an API or in the context of a Plan.

Product

Provide a method by which you can group APIs into a package that is intended for a particular use. Additionally, they contain Plans, which can be used to differentiate between different offerings. You can create Plans only within Products, and these Products are then published in a Catalog.

Provider Organization Owner

Owns and administers API provider organizations, manages application developer communities, authors APIs and defines products, manages the API product lifecycle. Owners are invited by the Cloud Administrator to join API Connect as an owner of a provider organization.

proxy

Application programming interface that forwards requests to a user-defined back-end resource and relays responses back to the calling application.

Region

A physical location (site or data center) hosting infrastructure isolated from other locations, with independent power and networking connectivity. A region may or may not have further sub-isolation characteristics such as semi-independent pods or availability zones.

Resources (User registries, TLS Profiles, Notifications)

Resources supply necessary functions for the API Connect cloud, such as user authentication, SSL security, and sending system-generated emails.

role

Defines permissions that can enable functionality for users. Each role has a different set of permissions.

security definition

Specifies all the settings for a particular aspect of API security; for example, the user registry that you use to authenticate access to the API.

server

A single appliance, such as an IBM WebSphere IBM DataPower appliance.

service

A user-configurable element implemented through one or more server processes in the API Connect runtime, such as microgateway, Analytics, IBM DataPower, and Developer Portal.

SNMP

Simple Network Management Protocol (SNMP) is a popular protocol for network management. It is used for collecting information from, and configuring, network devices, such as servers, printers, hubs, switches, and routers on an Internet Protocol (IP) network. The SNMP hosts are configured in the API Connect Cloud Manager.

Space

A subdivision of a Catalog. Each Space is used by a different API provider development team and has its own set of management capabilities relating specifically to the APIs that the associated team publishes to that Space, enabling each team to manage their APIs independently.

SSL (TLS) Profile

An SSL/TLS profile is used to secure the transmission of data through web sites. SSL certificates guarantee that information you submit to web sites will not be stolen or tampered with.

subscription

The means by which an application developer gains access to the resources provided by an API. An application developer uses the Developer Portal to subscribe to the plan in which the API is published.

TLS

Transport Layer Security - a cryptographic protocol that provides secure communication over a network to prevent eavesdropping and tampering. It is a successor to SSL and runs over TCP.

TLS Profiles

The Cloud Manager and API Manager use TLS profiles to secure transmission of data through web sites. TLS certificates guarantee that information you submit to web sites will not be stolen or tampered with. A TLS Profile consists of Server name, Protocol used (SSL or TLS), and whether Mutual Authentication is required.

Toolkit

See Developer Toolkit.

Topology administrator (role)

Configures gateway services for the Admin Organization.

Truststore

Stores certificates from trusted Certificate authorities(CA) which are used to verify certificate presented by Server in SSL/TSL Connection. While a keystore stores a server's credentials, the truststore stores certificates from a third-party CA.

User registry

A database or other collection containing credentials for users such as provider organization members, consumer organization members, and API Connect administrators. The users are authenticated by an identity provider such as LDAP. User registries authenticate users at login time when accessing the Cloud Manager or API Manager applications. User registries are also be used to protect APIs so that user credentials must be supplied when an API is called.

vendor extension

An extension to OpenAPI (Swagger) specification required by a particular use case.

Visibility

Setting visibility determines whether a Provider Organization has access to an Availability Zone, or other service.

Accessibility features for IBM API Connect

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in IBM® App Connect Enterprise. You can use screen-reader software to hear what is displayed on the screen.

- Supports keyboard-only operation.
- Supports interfaces commonly used by screen readers.

Tip: This product documentation, and its related publications, are accessibility-enabled for the IBM Home Page Reader. You can operate all features by using the keyboard instead of the mouse.

If you are reading a PDF file with a screen reader, the default reading option typically returns the best results. In some cases, how the PDF file is generated might require you to select one of the other reading options. For example, Use reading order in raw print stream.

Keyboard navigation

This product uses standard Linux® and Microsoft Windows navigation keys.

For more information about the commitments that IBM makes towards accessibility, see the [IBM Accessibility Center](#).

Legal information

Notices, and terms and conditions for information centers.

- [Tracking API volume for auditing and compliance](#)
For client security reasons, IBM entrusts its clients with monitoring their own API volume and ensuring that it is within the limits of the contract.
- [Notices](#)
This information was developed for products and services offered in the U.S.A.
- [Terms and conditions for information centers](#)
Permissions for the use of these publications are granted subject to the following terms and conditions.

Tracking API volume for auditing and compliance

For client security reasons, IBM entrusts its clients with monitoring their own API volume and ensuring that it is within the limits of the contract.

If you purchased IBM API Connect with a license metric of "API Calls", the count of total API Calls must be collected and archived by the client for audit and compliance. This includes, but is not limited to, licensed programs: "IBM API Connect Hybrid Entitlement" and "IBM Cloud Pak for Integration - API Calls".

Usage archiving

You **are required to regularly** record the number of your API calls from the Analytics **system** to maintain a record of **monthly** usage. To ensure the most accurate results, capture the counts of API Calls daily. For information about viewing API call counts and creating a query that counts API calls for **all** response status codes and for a specified time range, see [Counting total API calls for your Analytics service](#).

Overage charging

If you exceed your allotted usage, it is your responsibility to report this to IBM in the form of a CSV file so that the correct overage charge can be applied. IBM has the right to audit customer usage data at any time. If unreported overages are found, there are severe penalties.

Overage is based on the measurement period. For example, overage for a contract might be measured by the year. If your API volume exceeds the entitlement after 6 months, you must either pay overages for the remaining period or purchase additional volume.

Notices

This information was developed for products and services offered in the U.S.A.

IBM® may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive

Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14,
Shimotsuruma,
Yamato-shi
Kanagawa
242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England
SO21 2JN

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

Trademarks

IBM, the IBM logo, and `ibm.com`® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Copyright and trademark information page at <https://www.ibm.com/legal/copytrade.shtml>.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux® is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Kubernetes is a trademark of the Linux Foundation in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

OpenShift® is a trademark of Red Hat®, Inc. in the United States, other countries, or both.

VMware is a registered trademark or trademark of VMware, Inc. in the United States and other jurisdictions.

Other company, product, and service names might be trademarks of IBM or other companies.

Privacy Policy Considerations

IBM Software products, including software as a service solutions, (*Software Offerings*) may use cookies or other technologies to collect product usage information, to help improve the user experience, to tailor interactions with the user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth in the following paragraphs.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's session ID for purposes of session management, or functional purposes. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, See IBM's Privacy Policy at <https://www.ibm.com/privacy> and IBM's Online Privacy Statement at <https://www.ibm.com/privacy/details> the section entitled *Cookies, Web Beacons and Other Technologies* and the *IBM Software Products and Software-as-a-Service Privacy Statement* at <https://www.ibm.com/software/info/product-privacy>.

Terms and conditions for information centers

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM® website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the aforementioned instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM API Connect Considerations for GDPR Readiness

Information about features of IBM® API Connect that you can configure, and aspects of the product's use, that you should consider to help your organization with GDPR readiness.

For PID(s): 5725-Z22 5725-Z63

Notice:

This document is intended to help you in your preparations for GDPR readiness. It provides information about features of API Connect that you can configure, and aspects of the product's use, that you should consider to help your organization with GDPR readiness. This information is not an exhaustive list, due to the many ways that clients can choose and configure features, and the large variety of ways that the product can be used in itself and with third-party applications and systems.

Clients are responsible for ensuring their own compliance with various laws and regulations, including the European Union General Data Protection Regulation. Clients are solely responsible for obtaining advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulations that

may affect the clients' business and any actions the clients may need to take to comply with such laws and regulations.

The products, services, and other capabilities described herein are not suitable for all client situations and may have restricted availability. IBM does not provide legal, accounting, or auditing advice or represent or warrant that its services or products will ensure that clients are in compliance with any law or regulation.

Table of Contents

1. [GDPR](#)
2. [Product Configuration for GDPR](#)
3. [Data Life Cycle](#)
4. [Data Collection](#)
5. [Data Storage](#)
6. [Data Access](#)
7. [Data Processing](#)
8. [Data Deletion](#)
9. [Data Monitoring](#)
10. [Capability for Restricting Use of Personal Data](#)

GDPR

General Data Protection Regulation (GDPR) has been adopted by the European Union ("EU") and applies from May 25, 2018.

Why is GDPR important?

GDPR establishes a stronger data protection regulatory framework for processing of personal data of individuals. GDPR brings:

- New and enhanced rights for individuals
- Widened definition of personal data
- New obligations for processors
- Potential for significant financial penalties for non-compliance
- Compulsory data breach notification

Read more about GDPR

- [Complete guide to GDPR compliance](#)
- [IBM GDPR website](#)

Product Configuration - considerations for GDPR Readiness

The following sections provide considerations for configuring API Connect to help your organization with GDPR readiness.

Configuration to support data handling requirements

The GDPR legislation requires that personal data is strictly controlled and that the integrity of the data is maintained. This requires the data to be secured against loss through system failure and also through unauthorized access or via theft of computer equipment or storage media.

IBM API Connect stores identity data in a local database. This encompasses both clients' employee identity data and end users' identity data. Direct access to this database is not available. Identity information collected is protected in transit; refer to [Creating a TLS Server Profile](#) for details on configuring TLS profiles.

API Connect supports a variety of user registry types for authenticating users. Refer to [Authenticating by using your enterprise user directory](#) for details. When using a local user registry, passwords are stored in encrypted form in the local API Connect database. If you want alternative password management, leverage a non-user registry option to manage passwords.

Administrators, that you define, can view identity information. Administrators can take backups that include identity information. It is your responsibility to protect these backups.

A core component for an API Connect deployment is the gateway. Refer to the section on [API Gateways](#) (in the *API Connect components* topic) for details about gateways. DataPower® Gateways are commonly leveraged; refer to the DataPower Gateway documentation for details. Refer to the DataPower Gateway deployment guidelines document for considerations for configuring DataPower Gateways to help your organization with GDPR readiness.

Configuration to support Data Privacy

For Developer Portal, you can customize the privacy policy statement. Refer to [Customizing the privacy policy statement](#) for details.

Configuration to support Data Security

To learn about securing your solution, use this API Connect product documentation and search for "security".

Data Life Cycle

GDPR requires that personal data is:

- Processed lawfully, fairly and in a transparent manner in relation to individuals.
- Collected for specified, explicit and legitimate purposes.
- Adequate, relevant and limited to what is necessary.
- Accurate and, where necessary, kept up to date. Every reasonable step must be taken to ensure that inaccurate personal data are erased or rectified without delay.
- Kept in a form which permits identification of the data subject for no longer than necessary.

API Connect manages the life cycle of user-related data in the following ways:

User account information

API Connect collects and stores identity information, including first and last name, and email address, for the purposes of user registration. Cloud Manager and API Manager accounts are for your employees (or designated actors). Developer Portal accounts are for your consumers of your APIs. Identity information can be collected directly from users or can be copied from LDAP registries. In situations where non-local user registries are used, only the email address is copied from the LDAP registry. Developer Portal user accounts can be deleted; refer to [Deleting your Developer account](#) for details. Cloud Manager and API Manager user accounts' identity information can be anonymized by users.

Users of the API Manager UI can publish Products and APIs to the Developer Portal for Application Developers to access and use. Refer to [Developer Portal: socialize your APIs](#) to learn about Developer Portal. Developer Portal accounts are for consumers of your APIs. You can define and customize a terms and conditions statement that your users must accept before they can register to use your Developer Portal; refer to [Customizing the terms of use statement](#) for details.

API analytics

API Connect optionally logs information related to API invocations. This capability in API Connect is known as API Analytics. Refer to [API Analytics](#) for details. The API Analytics log information can optionally include unknown / unclassified information such as query headers and request and response information related to API calls -- you control defining the APIs and data associated with API invocations.

The retention period for storing Analytics data in API Connect is configurable; refer to [Configuring data retention and index rollover time periods](#) for details. Backup capability for this information is not available.

To disable API Analytics, disassociate it from the gateway that collects event data. For information, refer to [Associating an analytics service with a gateway service](#).

System logs

Event-logging preferences can be configured at the API level, refer to [Activity Log](#) for details.

API Connect logs collect technical information related to service use including tracing of service execution and sequences of operation use. Other technical data related to service use includes data values that define the mechanisms used to connect to the service, for example, IP address. This data is collected for debugging and service improvement. Service diagnostics are collected during unexpected or error situations to allow the offering team to correct the situation and hopefully prevent it from occurring in the future. There is no direct access available to these logs. You can configure log settings and download the logs as explained in [Logging](#).

API Connect can generate audit events. An audit event is logged from each management node when there are changes to the API lifecycle or to the organization; for example, publishing a product or creating an organization triggers this event. Refer to [Configuring audit logging to monitor user operations](#) for details.

Data Collection

Developer Portal accounts are for consumers of your APIs. You can define and customize a terms and conditions statement that your users must accept before they can register to use your Developer Portal; refer to [Customizing the terms of use statement](#) for details. You can also customize the privacy policy statement for Developer Portal; refer to [Customizing the privacy policy statement](#) for details.

Types of Data Collected

API Connect collects and stores identity information (including first name, last name, and email address) for the purposes of user registration. Cloud Manager and API Manager accounts are for your employees (or designated actors). Developer Portal accounts are for your consumers of your APIs. Identity information can be collected directly from users or can be copied from LDAP registries. In situations where non-local user registries are used, only the email address is copied from LDAP registry. Developer Portal user accounts can be deleted; refer to [Deleting your Developer account](#) for details. Cloud Manager and API Manager user accounts' identity information can be anonymized by users.

Data Storage

Identity data is stored in API Connect local data store. There is no direct access available to this data store.

API Analytics leverages the OpenSearch real-time distributed search and analytics engine for storage of logged data. There is no direct access available to this data store.

Identity data is included in backups; refer to [Backing up and restoring](#) for details on taking backups. It is your responsibility to protect and discard backups.

Data Access

Identity information can be viewed by administrators that you define.

Analytics information can be accessed through a variety of means. Refer to [Exporting API event data](#) and [Analytics in the Developer Portal](#) for details.

Analytics information can be offloaded to third-party systems. Refer to [Offloading analytics to third-party systems](#) for details. It is your responsibility to protect off-loaded data.

API Connect can generate audit events. An audit event is logged from each management node when there are changes to the API lifecycle or to the organization; for example, publishing a product or creating an organization triggers this event. Refer to [Configuring audit logging to monitor user operations](#) for details.

API Connect logs collect technical information related to service use including tracing of service execution and sequences of operation use. Other technical data related to service use includes data values that define the mechanisms used to connect to the service, for example, IP address. This data is collected for debugging and service improvement. Service diagnostics are collected during unexpected or error situations to allow the offering team to correct the situation and hopefully prevent it from occurring in the future. There is no direct access available to these logs. You can configure log settings and download the logs as explained in [Logging](#).

Data Processing

Data collected by API Connect or to gateways via API invocations is protected by TLS in transit. Refer to [Creating a TLS Server Profile](#) for details.

Data is stored in API Connect local database on the API Connect appliances. There is no direct access available to this data.

Cloud Manager and API Manager administrators (defined by you) have read access to identity data.

Data Deletion

Right to Erasure

Article 17 of the GDPR states that data subjects have the right to have their personal data removed from the systems of controllers and processors -- without undue delay -- under a set of circumstances.

Data Deletion characteristics

Users can delete their own Developer Portal user accounts -- refer to [Deleting your Developer account](#) for details. Cloud Manager and API Manager user account identity data can be anonymized by the users to remove association with the account data.

Technical information related to service use collected in logs is rolled over based on size and time criteria.

To disable API Analytics, disassociate it from the gateway that collects event data. For information, refer to [Associating an analytics service with a gateway service](#).

The retention period for storing Analytics data in API Connect is configurable; refer to [Configuring data retention and index rollover time periods](#) for details. Backup capability for this information is not available.

Analytics information can be offloaded to third-party systems. Refer to [Offloading analytics to third-party systems](#) for details. You are responsible for the security of off-loaded data.

Identity information for accounts is included in system backups. You manage the deletion of system backups.

Data Monitoring

Customers should regularly test, assess, and evaluate the effectiveness of their technical and organizational measures to comply with GDPR. These measures should include ongoing privacy assessments, threat modeling, centralized security logging, and monitoring

API Connect can generate audit events. An audit event is logged from each management node when there are changes to the API lifecycle or to the organization; for example, publishing a product or creating an organization triggers this event. Refer to [Configuring audit logging to monitor user operations](#) for details.

Capability for Restricting Use of Personal Data

Users of the API Manager UI can publish Products and APIs to the Developer Portal for Application Developers to access and use. Refer to [Developer Portal: socialize your APIs](#) to learn about Developer Portal. Developer Portal accounts are for consumers of your APIs.

You can define and customize a terms and conditions statement that your users must accept before they can register to use your Developer Portal; refer to [Customizing the terms of use statement](#) for details. You can also customize the privacy policy statement for Developer Portal; refer to [Customizing the privacy policy statement](#) for details.

Developer Portal users can modify their own account information, and can delete their account.

Essential reading

These articles by IBM® API Connect product specialists provide a wealth of supporting information on APIs and the API economy.

[Why Become a Digital Business?](#)

Executing a Digital Transformation to become a Digital Business is among the hottest initiatives now – crossing both business and IT. But, is this just the latest buzzword or is there something to this that is different? Do you know **why** you should become a “Digital Business”?

[Creating A Digital Ecosystem – Past, Present, and Future](#)

The ability to create a digital ecosystem is critical to digital transformation success. Your success is your network reach.

[Agile integration](#)

Your business needs a modern, agile approach to integration. It should empower extended teams to create integrations, leverages a complete set of integration styles and capabilities, and increases overall productivity.

[What is an API? and What is the API Economy?](#)

Businesses start to consider APIs and the API Economy at various times. Many are long down their API journeys, while others are still considering whether to start. This article looks at some of the basics that companies considering APIs might want to know.

[What is API Management?](#)

APIs are not new. Software and hardware have had APIs (Application Programming Interfaces) for decades. However, *having* an API and *managing* an API are not the same thing.

[Providing APIs or Managing APIs – There is a Big Difference](#)

A discussion of the potential for confusion between APIs that are provided and APIs that are managed.

[Recommendations for an API Economy Center of Excellence](#)

What roles are required to drive a successful API initiative, how should this fit in the current organization, and how do these roles relate to existing roles in the company?

[Focus on the API Developer](#)

A productive developer is a happy developer. One of the most frequently discussed topics in the API economy is focusing on the needs of the Application developer – the consumer target for your APIs.

[Agile API development](#)

Agile API Development Customer expectations and behavior are continuously changing. To deliver exceptional customer experience, a business must be nimble to adapt to these changing needs.

[API Products – Who, What, Where, When, Why and How”](#)

An API Product is an API offering made available for consumer use that is offered to a target market to satisfy a customer’s needs.

[Changing Culture – How Committed Are You?](#)

How do we change the culture in our organization to create an API culture?

[Plan Ahead! Don’t Build an API Superhighway into a Cul-de-sac](#)

Without proper planning, a business can start their API initiatives, build incredible excitement quickly, but find that the path they have taken leads them into a cul-de-sac (or dead end) that cannot handle the demand they have created.

[IBM API Connect v10.x Deployment WhitePaper](#)

A technical deep dive on the deployment options for IBM API Connect.

[Principles for API Security - White Paper](#)

API security is of paramount importance in gaining the promised benefits without exposure to negative consequences.

[Can you trust your APIs?](#)

As enterprises are continuously expanding their digital footprint, they must ensure the API behavior is intact, as it has a far-reaching effect on an application's execution and end-user experience.

[Istio Service Mesh and APIConnect/DataPower Gateway integration](#)

What is Istio, and how can DataPower API Gateway integrate in an Istio Service Mesh.

[Best Practice – Monetizing API Products](#)

Ultimately the goal for almost all API initiatives is Monetization in the broad sense that crosses multiple modes from Free - to Indirect - to Direct. Most businesses using APIs successfully are monetizing APIs by speeding offerings to market, reaching new markets and new customers, innovating, and sharing assets better inside their enterprise.

Tutorials

These tutorials guide you through the steps typically required to set up, configure and run an API Connect cloud.

Introduction

These tutorials are arranged according to the desired goal, such as creating and publishing an API, or managing catalogs of products, or consuming published API products. These goals can be grouped into categories as follows.

Table 1. Tutorial Categories

Category	Description
Cloud Administration	Setting up and configuring the API Connect cloud, including the creation of Provider Organizations.
API Provider Administration	Setting up and managing catalogs, inviting organization members (such as developers), creating Developer Portals, managing resources.
API Product Promotion and Usage	Managing the Developer Portal presentation and user administration, as well as consuming published API products as an external developer.

Time required

Each tutorial should take approximately 20 - 30 minutes to finish. If you explore other concepts that are related to this tutorial, it might take longer.

Use the links in the following sections to access the tutorials that fit your goals.

Cloud Administration

Table 2. Cloud Administration Tasks. Setting up and configuring the API Connect cloud, including the creation of Provider Organizations.

Category	Tutorials
Initial Configuration	<ul style="list-style-type: none">Initial Cloud Configuration
Provider Organization	<ul style="list-style-type: none">Creating a Provider Organization

API Provider Administration

Table 3. API Provider Tasks. Setting up and managing catalogs, inviting organization members (such as developers), creating Developer Portals, managing resources.

Category	Tutorials
Developer Portal Set Up	<ul style="list-style-type: none">Creating the PortalRegister a Consumer Application

API Product Promotion and Usage through the Developer Portal

There are many tutorials available for the Developer Portal from getting started to advanced customization. For more information, see [Developer Portal tutorials](#).

Prerequisites

- You must have a web browser available, whether you are working online or offline.
- When publishing Products in any of the tutorials you must have the permissions of an Organization Manager or Administrator. However, you can complete several of the tutorials with fewer permissions. For more information about user roles, see [Administering user access](#).

Installing and maintaining IBM API Connect

To ensure that your IBM® API Connect cloud functions, your cloud must have the necessary system requirements to support the installation. During the installation process, the components of IBM API Connect can be configured to satisfy your requirements.

Note: For a comprehensive technical guide to best practices, considerations, and deployment options for API Connect, see the [IBM API Connect v10.x Deployment WhitePaper](#).

- [Kubernetes, OpenShift, and Cloud Pak for Integration](#)

Use the instructions in this section to install, upgrade, and maintain API Connect on Kubernetes, OpenShift, and IBM Cloud Pak for Integration.

- [VMware](#)
Use the instructions in this section to install, upgrade, and maintain API Connect on VMware.
- [Migrating from API Connect v5 Public Cloud](#)
Prepare to move to API Connect v10 by migrating your existing data from API Connect v5 Public Cloud.
- [Migrating a Version 5 deployment](#)
Use the Migration Utility to automate the migration of a v5 deployment to a v10 deployment.
- [Migrating from v10 to v10 on a different form factor](#)
Migrate an API Connect v10 deployment from one form factor to a different form factor.
- [Migrating from v2018 to v10 on a different form factor](#)
Migrate an API Connect v2018 deployment to a different form factor on v10. For example, migrate v2018 on Kubernetes to v10 on OpenShift.
- [Replacing passwords, keys, and certificates](#)
Ensure the security of your API Connect deployment by replacing compromised passwords, keys, and certificates.
- [The API Connect operations tool: apicops](#)
Use the apicops operations tool to check the health, troubleshoot, and run maintenance operations on your API Connect installation.

Kubernetes, OpenShift, and Cloud Pak for Integration

Use the instructions in this section to install, upgrade, and maintain API Connect on Kubernetes, OpenShift, and IBM Cloud Pak for Integration.

- [Planning your deployment](#)
Review the following topics to plan your API Connect deployment.
- [Installing API Connect](#)
Use these instructions to install a deployment of API Connect on Kubernetes, OpenShift, and IBM Cloud Pak for Integration.
- [Upgrading API Connect](#)
You can upgrade a v10 deployment of API Connect to a newer version.
- [Maintaining API Connect](#)
You can use utilities to complete maintenance tasks such as backup, restore, and certificate management on Kubernetes.
- [Uninstalling API Connect](#)
You can uninstall a Kubernetes deployment of API Connect.

Planning your deployment

Review the following topics to plan your API Connect deployment.

- [Key concepts: Custom Resources, Operators, and Operands](#)
API Connect uses Custom Resources (CRs) and the Kubernetes Operator pattern.
- [Kubernetes ingress controller prerequisites](#)
Describes the prerequisite settings for the ingress controller for a Kubernetes runtime environment.
- [Persistent Volume \(PV\) requirements](#)
Review the persistent volumes (PVs) required by your API Connect installation.
- [Load balancer configuration in a Kubernetes deployment](#)
When deploying API Connect for High Availability, it is recommended that you configure a cluster with at least three nodes and a load balancer. A sample configuration is provided for placing a load balancer in front of your API Connect Kubernetes deployment.
- [Firewall requirements on Kubernetes](#)
Diagram for port configuration, and list of active ports, for an IBM® API Connect deployment on Kubernetes.
- [Planning your analytics deployment](#)
Plan your IBM API Connect analytics deployment by reviewing options for topology, data modifications, and storage.
- [Two data center deployment strategy on Kubernetes and OpenShift](#)
An overview of the two data center disaster recovery deployment strategy in API Connect.
- [Key Concepts: Cert-manager, Issuers, and Secrets](#)
By default API Connect uses an open source product that is called [cert-manager](#) to handle the issuing and renewal of the certificates that are used by API Connect. The cert-manager has its own Kubernetes pods and runs in its own namespace. The cert-manager adds some additional resources to the Kubernetes environment. The API Connect administrator needs to be familiar with these additional resources:
- [Certificates in a Kubernetes environment](#)
Use of cert-manager is recommended for managing certificates in a Kubernetes environment.
- [Webhooks deployed and managed by the API Connect Operator](#)
The API Connect Operator deploys and manages various Kubernetes API webhooks in the cluster to assist in management of its custom resources.

Key concepts: Custom Resources, Operators, and Operands

API Connect uses Custom Resources (CRs) and the Kubernetes Operator pattern.

- Kubernetes: <https://kubernetes.io/docs/concepts/extend-kubernetes/operator/>
- OpenShift: <https://developers.redhat.com/articles/2021/06/11/kubernetes-operators-101-part-1-overview-and-key-features>.

Below are some key points to understand about how API Connect implements this pattern:

- An API Connect instance is composed of two main operators and the API Connect subsystems that they manage, which in Kubernetes terminology are known as the Operands.
- The operator is responsible for monitoring the API Connect operands, ensuring that they are running as expected.
- API Connect operands can continue running without their operator, in an unmanaged state.

- The API Connect operator manages the Management Server, Portal, and Analytics subsystems. On Kubernetes each of these subsystems has its own Custom Resource, so each of these is an Operand. On OpenShift and Cloud Pak for Integration there is also the option of having a top-level Custom Resource which manages all subsystems as a single Operand.
- The DataPower operator manages the Gateway operand.
- A single API Connect operator can manage multiple API Connect instances if the operator is installed globally and the API Connect operands are in individual namespaces. This is the recommended configuration where multiple API Connect instances are required, as opposed to a separate operator in each namespace.
- There can be only one instance of the API Connect operator in your Kubernetes or OpenShift environment. You cannot have multiple namespaces each with their own API Connect operator.
- In a single OpenShift/K8S environment all API Connect instances, operands, and operators, in all namespaces must be at the same version. The only exception is during upgrade where the operators are upgraded first.

Planning your deployment topology

On Kubernetes, OpenShift, and Cloud Pak for Integration, API Connect provides a choice of deployment options for capacity, High Availability (HA), and Disaster Recovery (DR).

High availability versus disaster recovery

High availability focuses on minimizing the loss of service that is experienced by users during a hardware or software failure. Disaster recovery (DR) focuses on the procedures for recovering a system after a catastrophic hardware, software, or operator failure. Consider the following two metrics:

Recovery Time Objective (RTO)

The RTO is the time that it is acceptable for a system to be unavailable after a disaster.

Recovery Point Objective (RPO)

DR solutions are usually based on restoring from a backup. This means that the system can only be recovered to the state it was in when the last backup was taken, and not to the state it was in at the instant the disaster occurred. The RPO measures how far back in time the recovery point is, and therefore how much new data is lost. An RPO of zero would assert that no data is lost, but such a solution is often a compromise against the cost and performance of the system.

To achieve high availability in your API Connect deployment, the three replica deployment profiles should be used, where each API Connect component consists of clusters of three replicas that are distributed across three worker nodes. However, this topology requires that all cluster members are running in the same Kubernetes environment, which requires a low network latency between worker nodes that is often unachievable between geographically separated data centers. To overcome this limitation, API Connect provides a two data center deployment solution that has both a low Recovery Time Objective (RTO), and a low Recovery Point Objective (RPO).

Deployment options

API Connect is composed of three components: Management, Portal, and Analytics, and a supporting program: the DataPower Gateway, see [API Connect components](#). Note: Throughout this document, the DataPower Gateway supporting program is also referred to as a "component" or a "subsystem". The Management, Portal, and Analytics components are also referred to as "subsystems".

The Management, Portal, Analytics, and Gateway each have a set of capacity and HA profiles, called component profiles. OpenShift and Cloud Pak for Integration also have API Connect deployment profiles that define the component profiles that are used by each component of a complete API Connect deployment.

The naming convention of the profiles is **nAxcB.mC** where:

- **nA** indicates the number (A) of replicas of each pod, where each replica must run on a different worker node.
- **cB** indicates the minimum number of virtual cores (B) required on the worker node to deploy the profile.
- **mC** indicates the minimum amount of memory (C) in GB required by the profile on the worker node.

Component profiles

Each API Connect component has a defined set of profiles that it can be deployed with. For information on the resource settings defined for each component profile, see [Component profiles, CPU limits, memory limits, and licensing](#).

Key points of component profiles:

- One replica profiles are denoted by **n1**, and provide a single replica of each pod. The one replica profiles can be deployed on a single worker node.
- Three replica profiles are denoted by **n3**, and provide three replicas of each pod. At least three worker nodes must be available to deploy a three replica profile. If one replica pod fails, then the other two replicas continue serving requests, providing high availability.
- Component profiles do not have to be the same for each component. For example, you can have a three replica deployment of the Management component that is configured with a one replica deployment of the Portal component.
- It is possible to change component profile after installation see [Advanced configuration](#).
- Multiple Portal, Analytics, and Gateway component instances can be added with either configuration. For example, you can have a three replica Management component that is configured with two Portal services: a one replica Portal component, and a three replica Portal component.

OpenShift and Cloud Pak for Integration API Connect deployment profiles

API Connect deployment profiles define a complete API Connect deployment, specifying the component profiles for the Management, Portal, Analytics, and Gateway components to be used in that deployment.

Key points of API Connect deployment profiles:

- The API Connect deployment profile is selected by the user in the installation steps on OpenShift and Cloud Pak for Integration.
- API Connect deployment profiles provide one instance of the Portal, Analytics, and Gateway components.
- For details of each API Connect deployment profile, see: [API Connect deployment profiles for OpenShift and Cloud Pak for Integration](#)

Two data center disaster recovery

API Connect deployments can operate across two data centers in an active/warm-standby configuration. Each data center has a complete API Connect installation. The active data center processes all user operations. The warm-standby data center does not process user operations but maintains its management and portal databases in synchronization with the active data center. If a failure occurs at the active data center, the warm-standby data center can become active with a manual failover operation. For more information on the two data center disaster recovery configuration, see: [Two data center deployment strategy on Kubernetes and OpenShift](#)

API Connect deployment profiles for OpenShift and Cloud Pak for Integration

Details of the API Connect deployment profiles that are provided for OpenShift and Cloud Pak for Integration.

API Connect provides various profiles to suit your capacity and availability requirements.

The API Connect deployment profiles provide one Management component instance, and one instance of the Portal, Analytics, and Gateway components.

On Cloud Pak for Integration:

- The available deployment profiles are presented as tiles in the Platform UI.
- The Portal, Analytics, and Gateway components are automatically registered with the Management component during installation.

n1xc7.m48: Small - Single replica

Best suited for light, noncritical workloads such as small team development and testing. Deploys 1 replica of the Management, Developer Portal, Analytics, and Gateway components with a minimal footprint.

Table 1. n1xc7.m48

Component name	Component profile
Management	n1xc2.m16
Portal	n1xc2.m8
Analytics	n1xc2.m16
Gateway	n1xc1.m8

n1xc16.m72: Medium - Single replica

Ideal for medium workloads such as organizational development, testing, and production when in alignment with business availability requirements. Deploys 1 replica of the Management, Developer Portal, Analytics, and Gateway components.

Table 2. n1xc16.m72

Component name	Component profile
APIM	n1xc4.m16
Portal	n1xc4.m16
Analytics	n1xc4.m32
Gateway	n1xc4.m8

n3xc16.m48: Large - Three replicas

Ideal for heavy workloads when business requires redundancy for greater availability & capacity. Deploys 3 replicas of the Management, Developer Portal, Analytics, and Gateway components.

Table 3. n3xc16.m48

Component name	Component profile
APIM	n3xc4.m16
Portal	n3xc4.m8
Analytics	n3xc4.m16
Gateway	n3xc4.m8

n1xc12.m64: Medium - Single replica - Remote Gateway

Ideal for medium workloads such as organizational development, testing, and production when in alignment with business availability requirements. Must be configured for use with existing or independently deployed Gateways. Deploys 1 replica of the Management, Developer Portal, and Analytics components.

Table 4. n1xc12.m64

Component name	Component profile
APIM	n1xc4.m16
Portal	n1xc4.m16
Analytics	n1xc4.m32

n3xc12.m40: Large - Three replicas - Remote Gateway

Ideal for heavy workloads when business requires redundancy for greater availability & capacity. Must be configured for use with existing or independently deployed Gateways. Deploys 3 replicas the Management, Developer Portal, and Analytics components.

Table 5. n3xc12.m40

Component name	Component profile
----------------	-------------------

Component name	Component profile
APIM	n3xc4.m16
Portal	n3xc4.m8
Analytics	n3xc4.m16

Deprecated profiles

The API Connect deployment profiles presented here existed in previous releases, but are now in a deprecated state and should not be used for new installations.

Table 6. n3xc11.m48

Component name	Component profile
APIM	n3xc2.m16
Portal	n3xc4.m8
Analytics	n3xc4.m16
Gateway	n3xc1.m8

Table 7. n3xc14.m48

Component name	Component profile
APIM	n3xc2.m16
Portal	n3xc4.m8
Analytics	n3xc4.m16
Gateway	n3xc4.m8

Table 8. n1xc10.m48

Component name	Component profile
APIM	n1xc2.m16
Portal	n1xc2.m8
Analytics	n1xc2.m16
Gateway	n1xc4.m8

Component profiles, CPU limits, memory limits, and licensing

Available component profiles, their CPU and memory limits, and how the CPU-based licensing works.

API Connect is licensed based on the CPU usage of your deployment. Each component profile sets specific CPU and memory limits on each of the containers that comprise API Connect.

API Connect operator

The API Connect operator container uses the same CPU and memory requests and limits for all profiles, and is not a container that requires license entitlement.

Table 1. Operator container CPU and memory requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
ibm-apiconnect	ibm-apiconnect	75	1000	128	1Gi		1	No

The CPU and memory usage tables for each API Connect component profile are shown in the following pages.

- [Management component deployment profile limits](#)
The CPU, memory limits, and PVC space that are required for each Management profile.
- [Portal component deployment profile limits](#)
The CPU and memory limits that are defined for each Portal deployment profile.
- [Analytics component deployment profile limits](#)
The CPU and memory limits that are defined for each Analytics deployment profile.
- [DataPower gateway deployment profile limits](#)
The CPU and memory limits that are defined for each Gateway deployment profile.

Management component deployment profile limits

The CPU, memory limits, and PVC space that are required for each Management profile.

The tables in this topic also show which containers require license entitlement. The sum of the CPU limits of the licensed containers, multiplied by the number of replicas determine what you are charged. The formula is: $\langle \text{total licensed CPU limit} \rangle = \langle \text{total CPU limits of all licensed containers} \rangle * \langle \text{number of replicas} \rangle$. For example, for the `n3xc2.m16` profile:

- The three licensed containers are `apim`, `postgres`, and `taskmanager`.
- These three containers have CPU limits of 625, 750, and 625.
- This profile is a three replica (n3) profile.

The total CPU limit of the licensed containers then is: $(625 + 750 + 625) * 3 = 6000$

All Management component deployment profiles have job pods that complete tasks during startup, upgrade, backup, and restore operations. The job pods do not have any licensed containers, and their CPU/memory requests and limits are the same for all profiles.

Table 1. Management job pods CPU and memory requirements

Job Name	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits
management-up-apim-schema	management-up-apim-schema-0-to- <nnn>-<xx>-<yy>	apim-schema	50m	500m	128Mi	256Mi
management-up-apim-data-populate	management-up-apim-data-populate-0-to- <nnn>-<xx>-<yy>	apim-data-populate	50m	600m	256Mi	4096Mi
management-up-lur-schema	management-up-lur-schema-0-to- <nnn>-<xx>-<yy>	lur-schema	50m	500m	128Mi	256Mi
management-up-lur-data-populate	management-up-lur-data-populate-0-to- <nnn>-<xx>-<yy>	lur-data-populate	50m	500m	128Mi	256Mi
backrest-backup-management- <xx>-postgres	backrest-backup-management- <xx>-postgres- <yy>	backrest	not set	not set	not set	not set
management-apim-restore- <xxx>	management-apim-restore- <xxx>- <yyy>	apim-restore	50m	500m	128Mi	256Mi
management- <xxx>-postgres-rmdata	management- <xxx>-postgres-rmdata- <yyy>	rmdata	not set	not set	not set	not set
management- <xxx>-postgres-bootstrap	management- <xxx>-postgres-bootstrap- <yyy>	database	600m	750m	512Mi	8192Mi
management- <xxx>-postgres-stanza-create	management- <xxx>-postgres-stanza- create- <yyy>	backrest	50m	500m	128Mi	256Mi

The following tables show the pod resource and license requirement for each Management component deployment profile. A separate table shows the requirements for optional features.
Important: The tables for the three replica profiles show the usage for a single replica in the three replica deployment. Except for containers that are indicated to be single replica, multiply by three to get the total usage.

n1xc2.m16

Table 2. n1xc2.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
analytics-proxy	analytics-proxy	40m	800m	128Mi	256Mi		1	
apim	apim	200m	625m	128Mi	10Gi		1	Yes
client-downloads-server	client-downloads	20m	100m	8Mi	100Mi		1	
juhu	juhu	30m	800m	192Mi	512Mi		1	
ldap	ldap	20m	300m	64Mi	128Mi		1	
lur	lur	20m	500m	128Mi	256Mi		1	
natscluster	natscluster	40m	300m	16Mi	64Mi		1	
portal-proxy	portal-proxy	8m	100m	128Mi	256Mi		1	
postgres-backrest-shared-repo	pgbackrest	80m	200m	48Mi	128Mi	120Gi	1	
postgres	postgres	600m	750m	512Mi	8Gi	150Gi	1	Yes
pgbouncer	pgbouncer	80m	400m	24Mi	128Mi		1	
taskmanager	taskmanager	120m	625m	512Mi	10Gi		1	Yes
ui	ui-authentication	50m	50m	64Mi	64Mi		1	
	ui	20m	500m	64Mi	256Mi		1	
websocket-proxy	websocket-proxy	20m	100m	128Mi	256Mi		1	
postgres-operator	apiserver	40m	100m	128Mi	512Mi		1	
	operator	40m	100m	256Mi	256Mi		1	
	scheduler	20m	50m	128Mi	128Mi		1	
	event	20m	50m	128Mi	128Mi		1	

Table 3. n1xc2.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
ATM	hub	hub	200m	750m	512Mi	1Gi		1	Yes
	turnstile	turnstile	7m	50m	32Mi	64Mi		1	Yes
AI Test Gen (CP4i only)	openapi-analyzer	openapi-analyzer	500m	500m	2Gi	2Gi		1	Yes
	api-testgen-data	api-testgen-data	300m	300m	384Mi	384Mi		1	
	api-testgen-coordinator	api-testgen-coordinator	300m	300m	384Mi	384Mi		1	
	api-testgen-discovery	api-testgen-discovery	300m	300m	384Mi	384Mi		1	
	api-testgen-generator	api-testgen-generator	300m	300m	384Mi	384Mi		1	

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
Autotest assistant (CP4i only)	autotest-service	autotest-service	200m	200m	256Mi	256Mi		1	Yes
Billing	billing	billing	20m	300m	64Mi	196Mi/256Mi		1	Yes
2DCD R	postgres-remote	tunnel	30m	150m	512Mi	512Mi		1	
	tunnel	server	40m	500m	128Mi	256Mi		1	

n1xc3.m16

Table 4. n1xc3.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
analytics-proxy	analytics-proxy	40m	800m	128Mi	256Mi		1	
apim	apim	200m	900m	128Mi	10Gi		1	Yes
client-downloads-server	client-downloads	20m	100m	8Mi	100Mi		1	
juhu	juhu	30m	800m	192Mi	512Mi		1	
ldap	ldap	20m	300m	64Mi	128Mi		1	
lur	lur	20m	500m	128Mi	256Mi		1	
natscluster	natscluster	40m	300m	16Mi	64Mi		1	
portal-proxy	portal-proxy	8m	100m	128Mi	256Mi		1	
postgres-backrest-shared-repo	pgbackrest	80m	200m	48Mi	128Mi	120Gi	1	
postgres	postgres	600m	1200m	512Mi	8Gi	150Gi	1	Yes
pgbouncer	pgbouncer	80m	400m	24Mi	128Mi		1	
taskmanager	taskmanager	120m	900m	512Mi	10Gi		1	Yes
ui	ui-authentication	50m	50m	64Mi	64Mi		1	
	ui	20m	500m	64Mi	256Mi		1	
websocket-proxy	websocket-proxy	20m	100m	128Mi	256Mi		1	
postgres-operator	apiserver	40m	100m	128Mi	512Mi		1	
	operator	40m	100m	256Mi	256Mi		1	
	scheduler	20m	50m	128Mi	128Mi		1	
	event	20m	50m	128Mi	128Mi		1	

Table 5. n1xc3.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
ATM	hub	hub	200m	750m	512Mi	1Gi		1	Yes
	turnstile	turnstile	7m	50m	32Mi	64Mi		1	Yes
AISTestGen	openapi-analyzer	openapi-analyzer	500m	500m	2Gi	2Gi		1	Yes
	api-testgen-data	api-testgen-data	300m	300m	384Mi	384Mi		1	
	api-testgen-coordinator	api-testgen-coordinator	300m	300m	384Mi	384Mi		1	
	api-testgen-discovery	api-testgen-discovery	300m	300m	384Mi	384Mi		1	
	api-testgen-generator	api-testgen-generator	300m	300m	384Mi	384Mi		1	
Autotest assistant	autotest-service	autotest-service	200m	200m	256Mi	256Mi		1	Yes
	billing	billing	20m	300m	64Mi	196Mi/256Mi		1	Yes

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
2D CD R	postgres-remote	tunnel	30m	150m	512Mi	512Mi		1	
	tunnel	server	40m	500m	128Mi	256Mi		1	

n1xc4.m16

Table 6. n1xc4.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
analytics-proxy	analytics-proxy	100m	800m	128Mi	256Mi		1	
apim	apim	500m	1250m	128Mi	10Gi		1	Yes
client-downloads-server	client-downloads	50m	100m	8Mi	100Mi		1	
juhu	juhu	75m	800m	192Mi	512Mi		1	
ldap	ldap	50m	300m	64Mi	128Mi		1	
lur	lur	50m	500m	128Mi	256Mi		1	
natscluster	natscluster	100m	300m	16Mi	64Mi		1	
portal-proxy	portal-proxy	20m	100m	128Mi	256Mi		1	
postgres-backrest-shared-repo	pgbackrest	80m	200m	48Mi	128Mi	120Gi	1	
postgres	postgres	600m	1500m	512Mi	8Gi	150Gi	1	Yes
pgbouncer	pgbouncer	80m	400m	24Mi	128Mi		1	
taskmanager	taskmanager	300m	1250m	512Mi	10Gi		1	Yes
ui	ui-authentication	50m	50m	64Mi	64Mi		1	
	ui	50m	500m	64Mi	256Mi		1	
websocket-proxy	websocket-proxy	20m	100m	128Mi	256Mi		1	
postgres-operator	apiserver	100m	100m	128Mi	512Mi		1	
	operator	100m	100m	256Mi	256Mi		1	
	scheduler	50m	50m	128Mi	128Mi		1	
	event	50m	50m	128Mi	128Mi		1	

Table 7. n1xc4.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
AT M	hub	hub	500m	750m	512Mi	1Gi		1	Yes
	turnstile	turnstile	16m	50m	32Mi	64Mi		1	Yes
AI Te st Ge n	openapi-analyzer	openapi-analyzer	500m	500m	2Gi	2Gi		1	Yes
	api-testgen-data	api-testgen-data	300m	300m	384Mi	384Mi		1	
	api-testgen-coordinator	api-testgen-coordinator	300m	300m	384Mi	384Mi		1	
	api-testgen-discovery	api-testgen-discovery	300m	300m	384Mi	384Mi		1	
	api-testgen-generator	api-testgen-generator	300m	300m	384Mi	384Mi		1	
Auto test assis t	autotest-service	autotest-service	200m	200m	256Mi	256Mi		1	Yes
Bil lin g	billing	billing	50m	300m	64Mi	196Mi/256Mi		1	Yes
2D CD R	postgres-remote	tunnel	75m	150m	512Mi	512Mi		1	
	tunnel	server	100m	500m	128Mi	256Mi		1	

n3xc2.m16

Table 8. n3xc2.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
analytics-proxy	analytics-proxy	40m	800m	128Mi	256Mi		3	
apim	apim	200m	625m	128Mi	10Gi		3	Yes
client-downloads-server	client-downloads	20m	100m	8Mi	100Mi		3	
juhu	juhu	30m	800m	192Mi	512Mi		3	
ldap	ldap	20m	300m	64Mi	128Mi		3	
lur	lur	20m	500m	128Mi	256Mi		3	
natscluster	natscluster	40m	300m	16Mi	64Mi		3	
portal-proxy	portal-proxy	8m	100m	128Mi	256Mi		3	
postgres-backrest-shared-repo (single replica)	pgbackrest	80m	200m	48Mi	128Mi	180Gi	1	
postgres	postgres	600m	750m	512Mi	8Gi	230Gi	3	Yes
pgbouncer	pgbouncer	80m	400m	24Mi	128Mi		3	
taskmanager	taskmanager	120m	625m	512Mi	10Gi		3	Yes
ui	ui-authentication	50m	50m	64Mi	64Mi		3	
	ui	20m	500m	64Mi	256Mi		3	
websocket-proxy	websocket-proxy	20m	100m	128Mi	256Mi		3	
postgres-operator	apiserver	40m	100m	128Mi	512Mi		3	
	operator	80m	200m	256Mi	256Mi		3	
	scheduler	40m	100m	128Mi	128Mi		3	
	event	40m	100m	128Mi	128Mi		3	

Table 9. n3xc2.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
ATM	hub (single replica)	hub	200m	750m	1Gi	2Gi		1	Yes
	turnstile (single replica)	turnstile	40m	250m	128Mi	256Mi		1	Yes
AI Test Gen	openapi-analyzer (single replica)	openapi-analyzer	500m	500m	2Gi	2Gi		1	Yes
	api-testgen-data	api-testgen-data	300m	300m	384Mi	384Mi		1	
	api-testgen-coordinator	api-testgen-coordinator	300m	300m	384Mi	384Mi		1	
	api-testgen-discovery	api-testgen-discovery	300m	300m	384Mi	384Mi		1	
	api-testgen-generator	api-testgen-generator	300m	300m	384Mi	384Mi		1	
Autotest Assistant	autotest-service (single replica)	autotest-service	200m	200m	256Mi	256Mi		1	Yes
Billing	billing	billing	20m	300m	64Mi	196Mi/256Mi		3	Yes
2DCDR	postgres-remote	tunnel	30m	150m	512Mi	512Mi		1	
	tunnel	server	40m	500m	128Mi	256Mi		1	

n3xc3.m16

Table 10. n3xc3.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
analytics-proxy	analytics-proxy	40m	800m	128Mi	256Mi		3	
apim	apim	200m	900m	128Mi	10Gi		3	Yes
client-downloads-server	client-downloads	20m	100m	8Mi	100Mi		3	
juhu	juhu	30m	800m	192Mi	512Mi		3	

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
ldap	ldap	20m	300m	64Mi	128Mi		3	
lur	lur	20m	500m	128Mi	256Mi		3	
natscluster	natscluster	40m	300m	16Mi	64Mi		3	
portal-proxy	portal-proxy	8m	100m	128Mi	256Mi		3	
postgres-backrest-shared-repo (single replica)	pgbackrest	80m	200m	48Mi	128Mi	180Gi	1	
postgres	postgres	600m	1200m	512Mi	8Gi	230Gi	3	Yes
pgbouncer	pgbouncer	80m	400m	24Mi	128Mi		3	
taskmanager	taskmanager	120m	900m	512Mi	10Gi		3	Yes
ui	ui-authentication	50m	50m	64Mi	64Mi		3	
	ui	20m	500m	64Mi	256Mi		3	
websocket-proxy	websocket-proxy	20m	100m	128Mi	256Mi		3	
postgres-operator	apiserver	40m	100m	128Mi	512Mi		3	
	operator	80m	200m	256Mi	256Mi		3	
	scheduler	40m	100m	128Mi	128Mi		3	
	event	40m	100m	128Mi	128Mi		3	

Table 11. n3xc3.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
ATM	hub (single replica)	hub	200m	750m	1Gi	2Gi		1	Yes
	turnstile (single replica)	turnstile	40m	250m	128Mi	256Mi		1	Yes
AI Test Gen	openapi-analyzer (single replica)	openapi-analyzer	500m	500m	2Gi	2Gi		1	Yes
	api-testgen-data	api-testgen-data	300m	300m	384Mi	384Mi		1	
	api-testgen-coordinator	api-testgen-coordinator	300m	300m	384Mi	384Mi		1	
	api-testgen-discovery	api-testgen-discovery	300m	300m	384Mi	384Mi		1	
	api-testgen-generator	api-testgen-generator	300m	300m	384Mi	384Mi		1	
Autotest Assistant	autotest-service (single replica)	autotest-service	200m	200m	256Mi	256Mi		1	Yes
Billing	billing	billing	20m	300m	64Mi	196Mi/256Mi		3	Yes
2CDR	postgres-remote	tunnel	30m	150m	512Mi	512Mi		1	
	tunnel	server	40m	500m	128Mi	256Mi		1	

n3xc4.m16

Table 12. n3xc4.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
analytics-proxy	analytics-proxy	100m	800m	128Mi	256Mi		3	
apim	apim	500m	1250m	128Mi	10Gi		3	Yes
client-downloads-server	client-downloads	50m	100m	8Mi	100Mi		3	
juhu	juhu	75m	800m	192Mi	512Mi		3	
ldap	ldap	50m	300m	64Mi	128Mi		3	
lur	lur	50m	500m	128Mi	256Mi		3	
natscluster	natscluster	100m	300m	16Mi	64Mi		3	
portal-proxy	portal-proxy	20m	100m	128Mi	256Mi		3	

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
postgres-backrest-shared-repo (single replica)	pgbackrest	80m	200m	48Mi	128Mi	180Gi	1	
postgres	postgres	600m	1500m	512Mi	8Gi	230Gi	3	Yes
pgbouncer	pgbouncer	80m	400m	24Mi	128Mi		3	
taskmanager	taskmanager	300m	1250m	512Mi	10Gi		3	Yes
ui	ui-authentication	50m	50m	64Mi	64Mi		3	
	ui	50m	500m	64Mi	256Mi		3	
websocket-proxy	websocket-proxy	20m	100m	128Mi	256Mi		3	
postgres-operator	apiserver	100m	100m	128Mi	512Mi		3	
	operator	200m	200m	256Mi	256Mi		3	
	scheduler	100m	100m	128Mi	128Mi		3	
	event	100m	100m	128Mi	128Mi		3	

Table 13. n3xc4.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
ATM	hub (single replica)	hub	500m	750m	1Gi	2Gi		1	Yes
	turnstile (single replica)	turnstile	100m	250m	128Mi	256Mi		1	Yes
AI Test Gen	openapi-analyzer (single replica)	openapi-analyzer	500m	500m	2Gi	2Gi		1	Yes
	api-testgen-data	api-testgen-data	300m	300m	384Mi	384Mi		1	
	api-testgen-coordinator	api-testgen-coordinator	300m	300m	384Mi	384Mi		1	
	api-testgen-discovery	api-testgen-discovery	300m	300m	384Mi	384Mi		1	
Autotest Assistant	api-testgen-generator	api-testgen-generator	300m	300m	384Mi	384Mi		1	
	autotest-service (single replica)	autotest-service	200m	200m	256Mi	256Mi		1	Yes
Billing	billing	billing	50m	300m	64Mi	196Mi/256Mi		3	Yes
2DCDR	postgres-remote	tunnel	75m	150m	512Mi	512Mi		1	
	tunnel	server	100m	500m	128Mi	256Mi		1	

Portal component deployment profile limits

The CPU and memory limits that are defined for each Portal deployment profile.

The tables in this topic also show which containers require license entitlement. The sum of the CPU limits of the licensed containers, multiplied by the number of replicas determine what you are charged. The formula is: $\langle \text{total licensed CPU limit} \rangle = \langle \text{total CPU limits of all licensed containers} \rangle * \langle \text{number of replicas} \rangle$

All Portal component deployment profiles have a `sync-backup` job pod that completes tasks that are related to scheduled portal backups. The `sync-backup` job pod does not have any licensed containers, and its CPU/memory requests and limits are the same for all profiles.

Table 1. Portal job pods resource and licensing requirements

Job Name	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits
<code><portal-instance-name>-sync-backup</code>	<code><portal-instance-name>-sync-backup- <jobuid></code>	sync-ptl-backup-schedule	1m	1m	100Mi	100Mi

n1xc2.m8

Table 2. n1xc2.m8 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
db	db	200m	2000m	3Gi	8Gi	6Gi	1	
db	dbproxy	100m	1000m	128Mi	512Mi	15Gi	1	
www	web	150m	2000m	1Gi	5Gi	8Gi	1	Yes
www	admin	200m	2000m	1Gi	8Gi	6Gi + 15Gi	1	
nginx	nginx	20m	500m	512Mi	512Mi	1Gi	1	

Table 3. n1xc2.m8 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
2D CD R	db-remote	tunnel	75m	150m	160Mi	256Mi	1Gi	1	
	www-remote	tunnel	75m	150m	160Mi	256Mi	1Gi	1	
	tunnel	server	100m	500m	256Mi	512Mi	1Gi	1	

Note: For 2DCDR deployments, if the remote site is deployed with a three replica profile, the **db-remote** and **www-remote** pods have three replicas to match the remote site.

n1xc4.m16

Table 4. n1xc4.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
db	db	500m	3000m	6Gi	8Gi	6Gi	1	
	dbproxy	450m	1000m	256Mi	1Gi	15Gi	1	
www	web	600m	4000m	3Gi	8Gi	8Gi	1	Yes
	admin	800m	4000m	3Gi	8Gi	6Gi + 15Gi	1	
nginx	nginx	20m	500m	512Mi	512Mi	1Gi	1	

Table 5. n1xc4.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
2D CD R	db-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	1	
	www-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	1	
	tunnel	server	100m	500m	256Mi	512Mi	1Gi	1	

Note: For 2DCDR deployments, if the remote site is deployed with a three replica profile, the **db-remote** and **www-remote** pods have three replicas to match the remote site.

n1xc8.m16

Table 6. n1xc8.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
db	db	1300m	5000m	6Gi	8Gi	6Gi	1	
	dbproxy	1000m	1000m	256Mi	1Gi	15Gi	1	
www	web	1300m	8000m	3Gi	8Gi	8Gi	1	Yes
	admin	2300m	4000m	3Gi	8Gi	6Gi + 15Gi	1	
nginx	nginx	100m	500m	512Mi	512Mi	1Gi	1	

Table 7. n1xc8.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
2D CD R	db-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	1	
	www-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	1	
	tunnel	server	100m	500m	256Mi	512Mi	1Gi	1	

Note: For 2DCDR deployments, if the remote site is deployed with a three replica profile, the **db-remote** and **www-remote** pods have three replicas to match the remote site.

n3xc3.m8

Table 8. n3xc3.m8 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
db	db	400m	2000m	3Gi	8Gi	12Gi	3	
	dbproxy	150m	1000m	128Mi	512Mi	30Gi	3	
www	web	450m	3000m	1Gi	5Gi	12Gi	3	Yes
	admin	650m	3000m	1Gi	8Gi	12Gi + 30Gi	3	
nginx	nginx	20m	500m	512Mi	512Mi	1Gi	3	

Table 9. n3xc3.m8 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
2DCDR	db-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	3	
	www-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	3	
	tunnel	server	100m	500m	256Mi	512Mi	1Gi	3	

Note: For 2DCDR deployments, if the remote site is deployed with a one replica profile, the **db-remote** and **www-remote** pods have one replica, to match the remote site.

n3xc4.m8

Table 10. n3xc4.m8 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
db	db	500m	3000m	3Gi	8Gi	12Gi	3	
	dbproxy	450m	1000m	128Mi	512Mi	30Gi	3	
www	web	600m	4000m	1Gi	5Gi	12Gi	3	Yes
	admin	800m	4000m	1Gi	8Gi	12Gi + 30Gi	3	
nginx	nginx	20m	500m	512Mi	512Mi	1Gi	3	

Table 11. n3xc4.m8 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
2DCDR	db-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	3	
	www-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	3	
	tunnel	server	100m	500m	256Mi	512Mi	1Gi	3	

Note: For 2DCDR deployments, if the remote site is deployed with a one replica profile, the **db-remote** and **www-remote** pods have one replica, to match the remote site.

n3xc8.m16

Table 12. n3xc8.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
db	db	1300m	5000m	6Gi	8Gi	6Gi	3	
	dbproxy	1000m	1000m	256Mi	1Gi	15Gi	3	
www	web	1300m	8000m	3Gi	8Gi	8Gi	3	Yes
	admin	2300m	4000m	3Gi	8Gi	6Gi + 15Gi	3	
nginx	nginx	100m	500m	512Mi	512Mi	1Gi	3	

Table 13. n3xc8.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
2DCDR	db-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	3	
	www-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	3	
	tunnel	server	100m	500m	256Mi	512Mi	1Gi	3	

Note: For 2DCDR deployments, if the remote site is deployed with a one replica profile, the **db-remote** and **www-remote** pods have one replica, to match the remote site.

Analytics component deployment profile limits

The CPU and memory limits that are defined for each Analytics deployment profile.

The tables in this topic also show which containers require license entitlement. The sum of the CPU limits of the licensed containers, multiplied by the number of replicas determine what you are charged. The formula is: $\langle \text{total licensed CPU limit} \rangle = \langle \text{total CPU limits of all licensed containers} \rangle * \langle \text{number of replicas} \rangle$

All Analytics component deployment profiles have an **osinit** job pod that is run on startup and completes tasks that are related to initialization. The **osinit** job pod does not have any licensed containers, and its CPU/memory requests and limits are the same for all profiles.

You can choose different storage types with the three replica profiles, and so the pods deployed differ depending on the storage option chosen: [OpenSearch storage type](#). If you choose storage type **shared**, then your analytics deployment has 1 storage pod, called **storage**. If you choose storage type **dedicated**, then your analytics deployment has 2 storage pods, called **storage** and **storage-os-master**. The tables for the three replica profiles indicate in parentheses after the pod name which storage type the pod applies to, for example "storage (shared)".

Table 1. Analytics job pods

Job Name	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits
$\langle \text{analytics-instance-name} \rangle$ -osinit	$\langle \text{analytics-instance-name} \rangle$ -osinit- $\langle \text{podid} \rangle$	osinit	8 m	100 m	128 Mi	256 Mi

The PVC sizes that are specified in the tables are the defaults. Review the planning and installation documentation to decide on the sizes to allocate for your deployment: [Planning your analytics deployment](#).

n1xc2.m16

Table 2. n1xc2.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
mtls-gw	mtls-gw	100 m	1000 m	128 Mi	256 Mi		1	
ingestion	ingestion	275 m	1200 m	8 Gi	8 Gi	5 Gi	1	Yes
storage	storage	225 m	800 m	6 Gi	6 Gi	50 Gi	1	Yes
director	director	50 m	200 m	128 Mi	256 Mi		1	

n1xc4.m32

Table 3. n1xc4.m32 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
mtls-gw	mtls-gw	100 m	1000 m	256 Mi	512 Mi		1	
ingestion	ingestion	1300 m	2200 m	8 Gi	8 Gi	5 Gi	1	Yes
storage	storage	1200 m	1800 m	22 Gi	22 Gi	50 Gi	1	Yes
director	director	50 m	200 m	256 Mi	512 Mi		1	

n1xc6.m48

Table 4. n1xc6.m48 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
mtls-gw	mtls-gw	100 m	1000 m	256 Mi	512 Mi		1	
ingestion	ingestion	2000 m	3200 m	8 Gi	8 Gi	5 Gi	1	Yes
storage	storage	1800 m	2800 m	38 Gi	38 Gi	50 Gi	1	Yes
director	director	50 m	200 m	256 Mi	512 Mi		1	

n3xc4.m16

Table 5. n3xc4.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
mtls-gw	mtls-gw	100 m	1000 m	128 Mi	256 Mi		3	
ingestion	ingestion	1300 m	2200 m	8 Gi	8 Gi	5 Gi	3	Yes
director	director	50 m	200 m	128 Mi	256 Mi		3	
storage (shared)	storage	1200 m	1800 m	6 Gi	6 Gi	50 Gi	3	Yes
storage (dedicated)	storage	800 m	1000 m	4 Gi	4 Gi	50 Gi	3	Yes
storage-os-master (dedicated)	storage-os-master	400 m	800 m	2 Gi	2 Gi	5 Gi	3	Yes

n3xc4.m32

Table 6. n3xc4.m32 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
mtls-gw	mtls-gw	100 m	1000 m	256 Mi	512 Mi		3	
ingestion	ingestion	1300 m	2200 m	8 Gi	8 Gi	5 Gi	3	Yes
director	director	50 m	200 m	256 Mi	512 Mi		3	
storage (shared)	storage	1200 m	1800 m	22 Gi	22 Gi	50 Gi	3	Yes
storage (dedicated)	storage	800 m	1000 m	20 Gi	20 Gi	50 Gi	3	Yes
storage-os-master (dedicated)	storage-os-master	400 m	800 m	2 Gi	2 Gi	5 Gi	3	Yes

n3xc6.m48

Table 7. n3xc6.m48 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
mtls-gw	mtls-gw	100 m	1000 m	256 Mi	512 Mi		3	
ingestion	ingestion	2000 m	3200 m	8 Gi	8 Gi	5 Gi	3	Yes
director	director	50 m	200 m	256 Mi	512 Mi		3	
storage (shared)	storage	1800 m	2800 m	38 Gi	38 Gi	50 Gi	3	Yes

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
storage (dedicated)	storage	1400 m	1800 m	36 Gi	36 Gi	50 Gi	3	Yes
storage-os-master (dedicated)	storage-os-master	400 m	1000 m	2 Gi	2 Gi	5 Gi	3	Yes

n3xc8.m64

Table 8. n3xc8.m64 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
mtls-gw	mtls-gw	100 m	1000 m	256 Mi	512 Mi		3	
ingestion	ingestion	3000 m	4200 m	8 Gi	8 Gi	5 Gi	3	Yes
director	director	50 m	200 m	256 Mi	512 Mi		3	
storage (shared)	storage	2800 m	3800 m	54 Gi	54 Gi	50 Gi	3	Yes
storage (dedicated)	storage	2400 m	2800 m	50 Gi	50 Gi	50 Gi	3	Yes
storage-os-master (dedicated)	storage-os-master (dedicated)	400 m	1000 m	4 Gi	4 Gi	5 Gi	3	Yes

DataPower gateway deployment profile limits

The CPU and memory limits that are defined for each Gateway deployment profile.

The tables in this topic also show which containers require license entitlement. The sum of the CPU limits of the licensed containers, multiplied by the number of replicas determine what you are charged. The formula is: $\langle \text{total licensed CPU limit} \rangle = \langle \text{total CPU limits of all licensed containers} \rangle * \langle \text{number of replicas} \rangle$.

Note: All profiles show PVC size as 30Gi, this is only required if the Token Management Service (TMS) is enabled. If TMS is not enabled, there is no PVC requirement.

n1xc1.m8

Table 1. n1xc1.m8 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
gateway	datapower	1000m	1000m	8Gi	8Gi	30Gi	1	Yes

n1xc4.m8

Table 2. n1xc4.m8 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
gateway	datapower	4000m	4000m	8Gi	8Gi	30Gi	1	Yes

n3xc1.m8

Table 3. n3xc1.m8 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
gateway	datapower	1000m	1000m	8Gi	8Gi	30Gi	3	Yes

n3xc4.m8

Table 4. n3xc4.m8 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
gateway	datapower	4000m	4000m	8Gi	8Gi	30Gi	3	Yes

Network requirements for inter-subsystem communication

Plan how to secure the network communications between your API Connect subsystems.

API Connect provides various configuration options that are related to inter-subsystem communication.

Support load-balancer TLS termination by disabling mTLS and enabling JWT security

If your network infrastructure requires that load-balancers implement TLS termination, then mTLS between API Connect subsystems can be disabled and JSON Web Token (JWT) security can be used instead.

Note: Although mTLS is disabled, the network communication is still secured with standard TLS, which does not require passthrough to be enabled on the load-balancers. The following communication flows can be secured with JWT:

- Management initiated communication to portal, analytics, and gateway subsystems. From v10.0.5.3 and later.
- Analytics ingestion: The flow of API event records from the gateway to the analytics subsystem. From v10.0.5.4 and later.
Note: If the gateway is under a high transaction load and sending many API events to the analytics subsystem, the enablement of JWT might impact the performance of your analytics subsystem. It is recommended to run performance tests under your expected analytics load before you decide to use JWT for the analytics ingestion path.

With JWT enabled, the portal, gateway, and analytics subsystems verify the JSON Web Token (JWT) sent from the management subsystem when it initiates communication with them. The analytics subsystem verifies the JWT sent from the gateway with incoming API event data. The subsystems that receive the token contact a JSON Web Key Set (JWKS) URL to verify it. The JWKS URL is hosted by the management subsystem, in a subpath of the management subsystem's platform REST API. For more information about this feature, see [Enable JWT security instead of mTLS](#).

Use services instead of routes (OpenShift) or ingresses (Kubernetes)

Note: This feature is available on Kubernetes and OpenShift deployments where the API Connect subsystems are within the same cluster. It is not available on two data center disaster recovery deployments: [Two data center deployment strategy on Kubernetes and OpenShift](#).

For inter-subsystem communication, API Connect uses routes on OpenShift and Ingresses on Kubernetes. Both routes and ingresses require the exposure of external HTTPS endpoints. On API Connect deployments where all subsystems are in the same cluster, you can use services instead of routes or ingresses for inter-subsystem communication. When API Connect is configured to use services, all inter-subsystem communications are kept within the cluster, providing greater security and reduced load on external network infrastructure such as load-balancers.

In the context of inter-subsystem communication, the use of routes or ingresses is referred to as **external** communication, and the use of services is referred to as **in-cluster** communication.

You select whether to use **external** or **in-cluster** communication when you register your portal, gateway, or analytics subsystems with the management subsystem.

For more information about this feature, see [In-cluster service communication between subsystems](#).

Enabling mTLS on management to gateway communication

On Kubernetes and on OpenShift installations where the top-level CR is not used, mTLS can be enabled for management to gateway communication. To enable mutual TLS between the Management client and the Gateway service's manager endpoint, add `mtlsValidateClient` to the `spec` section:

```
spec:
  mtlsValidateClient: true
```

This ensures that the gateway service authenticates incoming requests from the API Manager, such as gateway service registration, and publishing APIs to the gateway service. Specifically, the gateway service requires that incoming requests present a certificate that was signed by the same CA that was used to sign the gateway service management endpoint. The gateway service management endpoint secret is specified under `gatewayManagerEndpoint.hosts.secretName`. The API Manager's gateway client's TLS credentials are specified in the ManagementCluster CR under `gateway.client.secretName`.

Note: In releases previous to 10.0.5.3, this property is called `validateApimClient`.

Note: On Cloud Pak for Integration and OpenShift where the top-level CR is used, mTLS enablement for management to gateway communication is not supported.

Management REST API CA certificate validation

The portal, gateway, and analytics subsystems make requests to the management subsystem through the REST API that is hosted by the management subsystem. This communication is standard TLS, where each REST API endpoint has a server certificate that is presented to the client during TLS handshaking. For extra security, validation of the management server certificate CA is enabled on all subsystems for new installations of API Connect v10.0.5.3 and later. If you are upgrading from an earlier release, the steps to enable this feature are

Certificates used to secure inter-subsystem communication

For information about the management of the certificates that are used in the TLS and mTLS communications between subsystems, see [Key Concepts: Cert-manager, Issuers, and Secrets](#).

Enable JWT security instead of mTLS

When and how to use JWT security instead of mTLS.

Most of the network interactions between API Connect subsystems use mTLS. The communication flows in API Connect that are secured with mTLS are as follows:

- Management to portal. mTLS set by default.
- Management to analytics. mTLS set by default.
- Management to API gateway. TLS set by default, mTLS can be enabled following [Enabling mTLS on management to gateway communication](#).
- Management to v5 compatible gateway. TLS set by default, mTLS can be enabled following [Enabling mTLS on management to gateway communication](#).
- Management to Event gateway. mTLS set by default. Cloud Pak for Integration only.
- API gateway to analytics. mTLS set by default.
- V5 compatible gateway to analytics. mTLS is set and cannot be disabled.
- Event gateway to analytics. mTLS set and cannot be disabled. Cloud Pak for Integration only.

All other inter-subsystem communication uses TLS. For the communication routes that use mTLS, it is a requirement that all network infrastructure such as load-balancers that are located between API Connect subsystems implement TLS passthrough (and not TLS termination).

Use of JSON Web Token (JWT) security instead of mTLS allows for the deployment of API Connect in network infrastructures where TLS termination is enabled on load-balancers that are located between subsystems. JWT provides application layer security between API Connect subsystems. The typical use-case for this feature is when payload inspection of all network communications is required at network boundaries. The communication flows that support JWT security are:

- Management to portal.
- Management to analytics.
- Management to API gateway.
- Management to v5 compatible gateway.
- API gateway to analytics. From v10.0.5.4 and later.

Note: It is not possible to use JWT and disable mTLS on the V5 compatible gateway to analytics message flow.

The management subsystem always sends a JWT when it initiates communication with the other subsystems, but by default the subsystems do not verify this JWT. When JWT security is enabled, the subsystems verify the JWT by making a REST call to the JSON Web Key Set (JWKS) URL. The JWKS URL is hosted on the management subsystem's platform REST API.

To enable JWT verification, you update the subsystem CRs to specify the JWKS URL. To disable JWT verification, you update the subsystem CRs and delete the value of the JWKS URL (leave it unset).

Note: For OpenShift users: The example steps in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place. If you are using a top-level CR you must edit the API Connect cluster CR (the top-level CR), instead of directly in the subsystem CRs. If the subsystem section is not included in the top-level CR, copy and paste the section from the subsystem CR to the top-level CR.

Disabling mTLS and enabling JSON Web Token (JWT) security

If you disable mTLS, you must enable JWT. It is not possible to configure API Connect with both mTLS and JWT disabled.

You disable mTLS on the portal, gateway, and analytics subsystems by setting the `mtlsValidateClient` property to false in the Custom Resources (CRs) of each subsystem.

Note: By default, `mtlsValidateClient` is set to false on the gateway subsystem.

When `mtlsValidateClient` is set to false, the subsystem does not verify the client's TLS credentials, but it does check for a valid JWT configuration. For example, to disable mTLS and enable JWT for management to portal communication, edit the portal CR and set:

```
spec:
  mtlsValidateClient: false
  jwksUrl: <JWKS URL>
```

where `<JWKS URL>` is the `jwksUrl` specified in the management subsystem CR `status.endpoints` section.

Using registration of the portal service in the Cloud Manager UI as an example, JWT works as follows:

1. Management subsystem establishes a TLS connection to the portal subsystem on the portal director admin endpoint, and sends a JWT to the portal.
2. Before the portal trusts the management subsystem, it first verifies the JWT it received. The JWT is verified by a call to a JSON Web Key Set (JWKS) URL.
3. The management subsystem receives the JWT verification request from the portal on its platform REST API, and responds that the JWT is valid.
4. The portal receives confirmation that the JWT is valid, and so it can trust the management subsystem and continue the registration process.

Note:

Datapower version 10.5.0.4 only: To enable JWT on the gateway to analytics communications flow, you also must add a `dataPowerOverride` section to the gateway CR:

```
spec:
  jwksUrl: <JWKS URL>
  dataPowerOverride:
    image: customregistry.com/custom-image-datapower:10.5.0.5
    version: 10.5.0.5
    license: X-XXXX-XXXXXX
```

To disable JWT security, either remove the `jwksUrl` property from the CR (and the `dataPowerOverride` if using gateway v10.5.0.4), or leave its value empty. If you disable JWT, you must enable mTLS by setting `mtlsValidateClient` to true.

Configuring the JWKS URL

The JWKS URL is defined as the platform API hostname with the following subpath appended: `api/cloud/oauth2/certs`. You can see what the JWKS URL is from the `status.endpoints` section of the management CR, for example:

```
status:
  ...
  endpoints:
  ...
  - name: jwksUrl
    secretName: api-endpoint
    type: API
    uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

Note: The JWT feature is intended for environments where there are load-balancers located between subsystems. If you are using the in-cluster communications feature [Use services instead of routes \(OpenShift\) or ingresses \(Kubernetes\)](#), then you cannot have load-balancers between subsystems, and so disabling mTLS and enabling JWT is not necessary. However, if you do want to use the JWT feature with in-cluster communications then the JWKS URL can be obtained following these steps:

1. Describe any API Connect pod:

```
kubectl describe pod <apic-pod>
```

2. In the pod describe output, search for `APIC_PLATFORM_API_SVC_ENDPOINT`:

```
Containers:
  apim:
    Environment:
      APIC_PLATFORM_API_SVC_ENDPOINT: https://management-juhu.acme-v1005-020323.svc:2000/api
```

- The JWKS URL is the value of `APIC_PLATFORM_API_SVC_ENDPOINT` with `/cloud/oauth2/certs` appended to it. If the `APIC_PLATFORM_API_SVC_ENDPOINT` is missing the `/api` at the end, add `api/cloud/oauth2/certs` instead. The path of the JWKS URL must always be `api/cloud/oauth2/certs`.

Key points and limitations of JWT security

- For API Connect deployments upgraded from a pre-v10.0.5.3 release, the behavior of each subsystem remains the same as it was before upgrade.
- Changing the JWKS URL on portal, and analytics subsystems causes pods to restart, and so a short subsystem outage. The gateway pod does not restart when the JWKS URL is changed.
- On new installs of Cloud Pak for Integration and top-level CR OpenShift deployments, the JWKS URL is set automatically for the portal, gateway, and analytics subsystems.
- You can have both mTLS and JWT enabled, although the analytics subsystem does not verify the JWT if mTLS is enabled.
- The v5c gateway does not send a JWT to the analytics subsystem. This flow is secured by mTLS, which cannot be disabled.
- If you enable JWT for gateway to analytics communication, the ingestion of analytics data becomes dependent on the availability of the management subsystem (so that the JWT can be verified). For more information about this limitation, see: [JWT usage between gateway and analytics subsystems](#).
- Under high analytics ingestion loads, JWT enablement might impact the performance of the analytics subsystem. It is recommended to complete a performance test under your expected analytics load before you decide to use JWT for gateway to analytics communication.

JWT usage between gateway and analytics subsystems

Because the analytics subsystem makes a REST call to the management subsystems to verify the JWT it receives from the gateway, analytics ingestion depends on the management subsystem when JWT is enabled. The analytics subsystem caches the response from the JWKS URL so that it does not need to make a REST call to the management subsystem every time API event data is received from the gateway. But the JWT verification cache does expire after 24 hours, and a new REST call to the management subsystem is made to refresh the cache. The JWT dependency on the platform REST API means that an outage of the management subsystem does eventually prevent the analytics subsystem from accepting new analytics API event records.

In-cluster service communication between subsystems

Key points and limitations of `in-cluster` inter-subsystem communication.

- In-cluster communication is only possible between subsystems that are in the same cluster.
- In-cluster communication cannot be used in two data center disaster recovery deployments, [Two data center deployment strategy on Kubernetes and OpenShift](#).
- If you are upgrading from a pre-10.0.5.3 release, all existing subsystems continue to use `external` communication. If you want to change upgraded subsystems to use `in-cluster`, then you must reregister them.
- If you are adding new subsystems to an upgraded deployment you can set the subsystems to use `in-cluster` communication, but you must use different certificates and secrets for the subsystem endpoints. The default certificate and secret names for the subsystem endpoints are:
 - Analytics: `ai-endpoint`.
 - Portal: `portal-admin`.
 - Gateway: `gmv6-manager-endpoint` or `gw-gateway-manager`

Do not use these same certificate and secret names if your additional subsystems are in the same namespace.

- After a portal, gateway, or analytics subsystem is registered with the management subsystem, to change the communication mechanism you must reregister the subsystem. For more information on this procedure, see:
 - [Kubernetes: Change inter-subsystem communication type](#).
 - [OpenShift: Change inter-subsystem communication type](#).

Important: Backups of the management subsystem cannot be restored if the communication type of any registered subsystem is changed after the backup was taken. Do not change the communication type of any of your subsystems if you might want to restore your management subsystem from a previous backup.

- If you customize any TLS certificates used for inter-subsystem communication, then to use `in-cluster` communication the TLS certificates must include the service hostname in the DNS section of the SAN, for example:

```
x509v3 Subject Alternative Name: critical
DNS: ptladmin.mydomain.com, DNS: portal.apic.svc, DNS: portal.apic.svc.cluster.local
```

- On Cloud Pak for Integration, all subsystems are registered automatically during deployment with `external` communication specified.

Kubernetes ingress controller prerequisites

Describes the prerequisite settings for the ingress controller for a Kubernetes runtime environment.

Before you begin

Note: This article refers to third-party software that IBM does not control. As such, the software may change and this information may become outdated.

These instructions assume you have a working Kubernetes environment and understand how to manage Kubernetes. Kubernetes is a platform for automated deployment, scaling, and operation of application containers across clusters of hosts, providing container-centric infrastructure. For more information, see <https://kubernetes.io>.

Kubernetes/ingress-nginx ingress controller `ingress-config.yaml` settings

A Kubernetes deployment for IBM® API Connect requires the `kubernetes/ingress-nginx` ingress controller implementation (see <https://github.com/kubernetes/ingress-nginx>) with SSL passthrough enabled.

API Connect v10 does not require Helm, so it is recommended to use Helm3 for the installation of the ingress controller. Follow these steps:

- Create a file `ingress-config.yaml` where the following values are required:
 - Specify at least one SSL protocol; separate multiple protocols with a comma as shown in the example.

```
controller:
  watchIngressWithoutClass: true
  admissionWebhooks:
```

```

    enabled: false
  config:
    ssl-protocols: "TLSv1.2 TLSv1.3"
  extraArgs:
    annotations-prefix: ingress.kubernetes.io
    enable-ssl-passthrough: true

```

You may use the following sample ingress-config.yml file to configure the ingress controller:

```

controller:
  watchIngressWithoutClass: true
  admissionWebhooks:
    enabled: false
  config:
    hsts-max-age: "31536000"
    keepalive: "32"
    log-format: '{ "@timestamp": "$time_iso8601", "@version": "1", "clientip": "$remote_addr",
    "tag": "ingress", "remote user": "$remote user", "bytes": $bytes_sent, "duration":
    $request_time, "status": $status, "request": "$request_uri", "urlpath": "$uri",
    "urlquery": "$args", "method": "$request_method", "referer": "$http_referer",
    "useragent": "$http_user_agent", "software": "nginx", "version": "$nginx_version",
    "host": "$host", "upstream": "$upstream_addr", "upstream-status": "$upstream_status"
    }'
    main-snippets: load_module "modules/nginx_stream_module.so"
    proxy-body-size: "0"
    proxy-buffering: "off"
    server-name-hash-bucket-size: "128"
    server-name-hash-max-size: "1024"
    server-tokens: "False"
    ssl-ciphers: HIGH:!aNULL!MD5
    ssl-prefer-server-ciphers: "True"
    ssl-protocols: "TLSv1.2 TLSv1.3"
    use-http2: "true"
    worker-connections: "10240"
    worker-cpu-affinity: auto
    worker-processes: "1"
    worker-rlimit-nofile: "65536"
    worker-shutdown-timeout: 5m
  daemonset:
    useHostPort: false
  extraArgs:
    annotations-prefix: ingress.kubernetes.io
    enable-ssl-passthrough: true
  hostNetwork: true
  kind: DaemonSet
  name: controller
rbac:
  create: "true"

```

2. Run the commands:

```

helm3 repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm3 repo update
helm3 install ingress-controller ingress-nginx/ingress-nginx --namespace kube-system --values ingress-config.yaml

```

Kubernetes/ingress-nginx ingress controller `config.map` settings

To ensure that the IBM API Connect services have time to start, increase the `proxy-read-timeout` and `proxy-send-timeout` values, which are in seconds, in the `kubernetes/ingress-nginx` ingress controller `config.map` to at least the following:

- `proxy-read-timeout: "240"`
- `proxy-send-timeout: "240"`

Depending on your environment, you might need to increase these further if the IBM API Connect services do not start. If there is a load balancer in front of the worker nodes, then the load balancer configuration might also need to have extended timeouts.

System and Software Requirements

The system and software requirements are described in the Software Product Compatibility Reports. See [Detailed system requirements for a specific product](#)

Persistent Volume (PV) requirements

Review the persistent volumes (PVs) required by your API Connect installation.

The PV requirements for API Connect subsystems vary according to the number of subsystem nodes that are deployed, whether you are using two datacenter disaster recovery (2DCDR) [Two data center deployment strategy on Kubernetes and OpenShift](#), and other subsystem specific options.

Management subsystem

Table 1. Management PV requirements

Deployment profile	Number of PVs required
Single node (n1)	4
Three node (n3)	10

Note: An additional 4 PVs are required for upgrade of the n3 deployment profile. Replaced PVs can be deleted after successful upgrade.

Gateway subsystem

Only required where `tokenManagementService` is enabled:

```

tokenManagementService:
  enabled: true
  storage:
    storageClassName: $STORAGE_CLASS
    volumeSize: 30Gi

```

Portal subsystem

Table 2. Portal PV requirements

Deployment profile	Number of PVs required
Single node (n1), one datacenter	6
Three nodes (n3), one datacenter	18
Single node (n1), two datacenter	9 per datacenter
Three nodes (n3), two datacenter	27 per datacenter
Single node (n1), two datacenter, where other datacenter is three node (n3)	13 in n1 datacenter, 23 in n3 datacenter
Three node (n3), two datacenter, where other datacenter is single node (n1)	30 in n3 datacenter, 15 in n1 datacenter

Analytics subsystem

The number of PVs needed varies according to whether the [persistent queue](#) feature is used, and if internal storage is enabled [Planning to disable internal storage](#).

Table 3. Analytics PV requirements

Deployment profile	Storage enabled	Storage disabled (ingestion only)	Extra PVs if persistent queue enabled
Single node (n1)	1	0	1
Three node (n3)	3	0	3

Note: The last column shows the PVs that must be added to those specified in the storage columns. For example, if you have an n1 profile with storage and persistent queue enabled, then you need 2 PVs.

Load balancer configuration in a Kubernetes deployment

When deploying API Connect for High Availability, it is recommended that you configure a cluster with at least three nodes and a load balancer. A sample configuration is provided for placing a load balancer in front of your API Connect Kubernetes deployment.

About this task

API Connect can be deployed on a single node cluster. In this case the ingress endpoints are host names for which the DNS resolution points to the single IP address of the corresponding node hosting a particular subsystem, and no load balancer is required. For high availability, it is recommended to have at least a three node cluster. With three nodes, the ingress endpoints cannot resolve to a single IP address. A load balancer should be placed in front of an API Connect subsystem to route traffic.

Because it is difficult to add nodes once endpoints are configured, a good practice is to configure a load balancer even for single node deployments. With the load balancer in place, you can easily add nodes when needed. Add the node to the list of servers pointed to by the load balancer and the ingress endpoints defined during installation of API Connect can remain unchanged.

To support Mutual TLS communication between the API Connect subsystems, configure the load balancer with **SSL Passthrough** and **Layer 4** load balancing. In order for Mutual TLS to be performed directly by the API Connect subsystems, the load balancer should leave the packets unmodified, as is accomplished by Layer 4. Following is a description of the communication between the endpoints that are configured with Mutual TLS:

- API Manager (with the client certificate portal-client) communicates with the Portal Admin endpoint portal-admin (with the server certificate portal-admin-ingress)
- API Manager (with the client certificate analytics-ingestion-client) communicates with the Analytics Ingestion endpoint analytics-ingestion (with the server certificate analytics-ingestion-ingress)

Note: From v10.0.5.3, it is possible to disable mTLS and use JWT instead, which allows the load-balancers to do TLS termination. For more information, see [Enable JWT security instead of mTLS](#).

Set endpoints to resolve to the load balancer

When configuring a load balancer in front of the API Connect subsystems, the ingress endpoints are set to host names that resolve to a load balancer, rather than to the host name of any specific node. For an overview of endpoints, see [Deployment overview for endpoints and certificates](#).

Use these example topologies as a guideline to determine the best way to configure the load balancer for your deployment.

Procedure

• Kubernetes cluster with wildcard DNS

In this example, API Connect is deployed to a Kubernetes cluster with three master and six worker nodes. The master nodes have the Kubernetes API server listening on port 6443. The Kubernetes API Server is used for communicating with the kubectl command line interface. Note that it is generally not necessary for the Kubernetes API server (on port 6443 in this example) to be exposed outside the cluster through the load balancer, this is just one possible setup shown here.

The kubernetes/ingress-nginx ingress controller is deployed as a daemonset, so that every worker node in the cluster has an ingress controller pod listening on port 443. The API Connect subsystems (API Manager, Developer Portal, Analytics and Gateway) are all deployed on this same cluster.

Note:

- When you configure a load balancer in front of a Management subsystem, specify timeouts of at least 240 seconds. Note that large deployments might need larger values.

The default timeout is typically 50 or 60 seconds, which is not long enough to avoid **409**

Conflict or **504 Gateway Timeout** errors. The **409 Conflict** error can occur when the time needed to complete an operation is sufficiently long that a

second request gets issued.

For example, to specify 240 seconds when using HAProxy as a load balancer, set `timeout client` and `timeout server` to 240000.

Best practice is to ensure that the same values are specified for the timeout settings for the load balancer and for the Kubernetes ingress controller. The ingress controller timeout settings are set in the ingress controller config.map. For more information, see the `proxy-read-timeout` and `proxy-send-timeout` settings in [Kubernetes ingress controller prerequisites](#).

- Ensure that the load balancer is configured to support the cipher `TLS_ECDHE_RSA_WITH_AES256_GCM_SHA384 (0xc030)`. When the Developer Portal, as a client, calls back to `platform-api-endpoint` or `consumer-api-endpoint` of the management subsystem, it sends only a single cipher in the Client Hello. The cipher is `TLS_ECDHE_RSA_WITH_AES256_GCM_SHA384 (0xc030)`. The SSL Handshake will fail if this cipher is not configured (supported) on the client's load balancer.

A host running HAProxy acts as the load balancer, with a configuration for proxies in the HAProxy configuration file such as:

```
defaults
    log          global
    mode        http
    option      httplog
    option      dontlognull
    timeout connect 5000
    timeout client 240000
    timeout server 240000
    errorfile 400 /etc/haproxy/errors/400.http
    errorfile 403 /etc/haproxy/errors/403.http
    errorfile 408 /etc/haproxy/errors/408.http
    errorfile 500 /etc/haproxy/errors/500.http
    errorfile 502 /etc/haproxy/errors/502.http
    errorfile 503 /etc/haproxy/errors/503.http
    errorfile 504 /etc/haproxy/errors/504.http

frontend apiservers
    bind *:6443
    mode tcp
    option tcplog
    option forwardfor
    default_backend k8s_apiservers

frontend ingress
    bind *:443
    mode tcp
    option tcplog
    option forwardfor
    default_backend k8s_ingress

backend k8s_apiservers
    mode tcp
    option tcplog
    option ssl-hello-chk
    option log-health-checks
    default-server inter 10s fall 2
    server cluster1-master-1 10.169.241.226:6443 check
    server cluster1-master-2 10.169.241.163:6443 check
    server cluster1-master-3 10.169.241.153:6443 check

backend k8s_ingress
    mode tcp
    option tcplog
    option ssl-hello-chk
    option log-health-checks
    default-server inter 10s fall 2
    server cluster1-worker-1 10.169.241.142:443 check
    server cluster1-worker-2 10.169.241.150:443 check
    server cluster1-worker-3 10.169.241.188:443 check
    server cluster1-worker-4 10.169.241.196:443 check
    server cluster1-worker-5 10.169.241.168:443 check
    server cluster1-worker-6 10.169.241.156:443 check
```

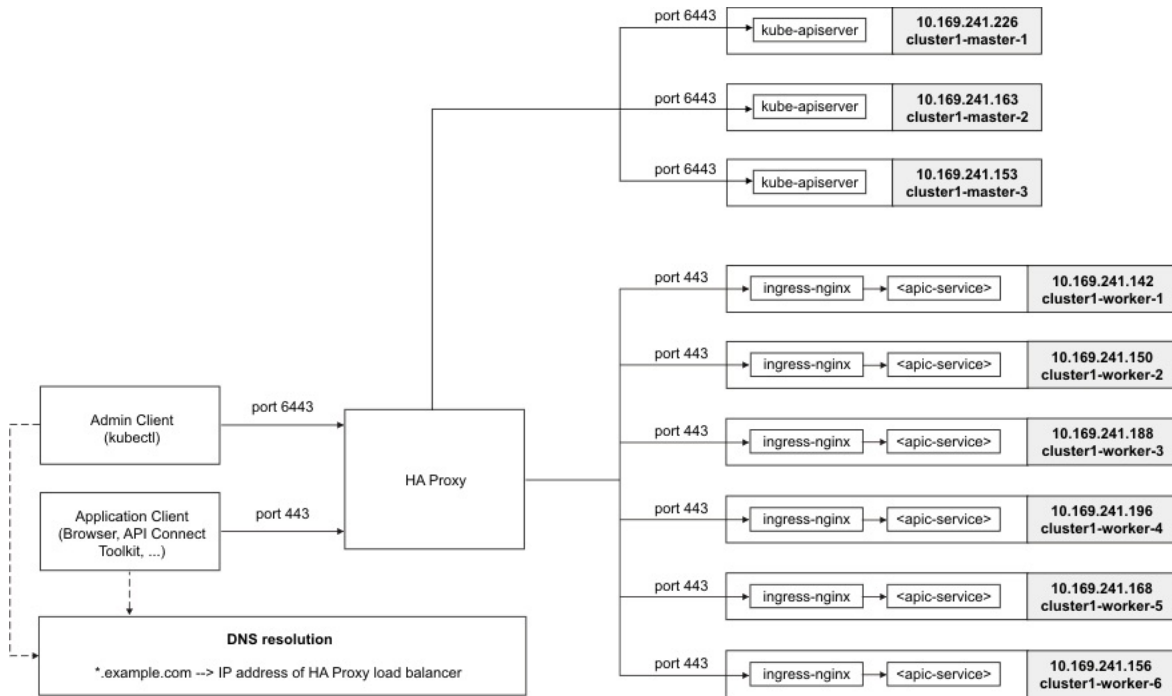
A wildcard DNS record allows the resolution of `*.example.com` to the IP address of the HAProxy host.

The ingress endpoints are defined in the CRs (custom resources) for each API Connect subsystem. During configuration of each subsystem, you define the endpoint by specifying a host name `$$STACK_HOST` in the CR. For configuration instructions, see [Installing the API Connect subsystems](#).

You can view the endpoint definitions in sample CRs:

- [Installing the Management subsystem cluster](#)
- [Installing the Developer Portal subsystem](#)
- [Installing the Analytics subsystem](#)
- [Installing the DataPower Gateway subsystem](#)

The following diagram illustrates the wildcard DNS example:



- **Kubernetes cluster without wildcard DNS**

This example uses the same topology as the previous one, except that there is no wildcard DNS resolution. Instead, individual DNS records are added, all pointing to the IP address of the load balancer. For example: `portal.example.com` and `admin.portal.example.com` resolve to the IP address of the load balancer.

- **Kubernetes cluster with `xip.io` or `nip.io`**

This example uses the same topology as the first one, except that there is no specific DNS resolution configured and instead a service such as `xip.io` or `nip.io` provides wildcard DNS resolution. It is not recommended to use this approach for a production setup, as the ingress endpoints configured during installation of API Connect would be tied to the IP address of the load balancer. This approach can be useful when configuring a test deployment in situations where access to DNS records is not practical or would introduce delay in the deployment.

Firewall requirements on Kubernetes

Diagram for port configuration, and list of active ports, for an IBM® API Connect deployment on Kubernetes.

Required Ports between zones

The following network diagram example helps to explain which ports must be configured in an API Connect network. Specific ports must be configured to enable the communication between the various zones, both public and private, in a network.

The ports specified in the diagram are default ports. Check your deployment to understand which communication, if any, is configured to use non-default ports.

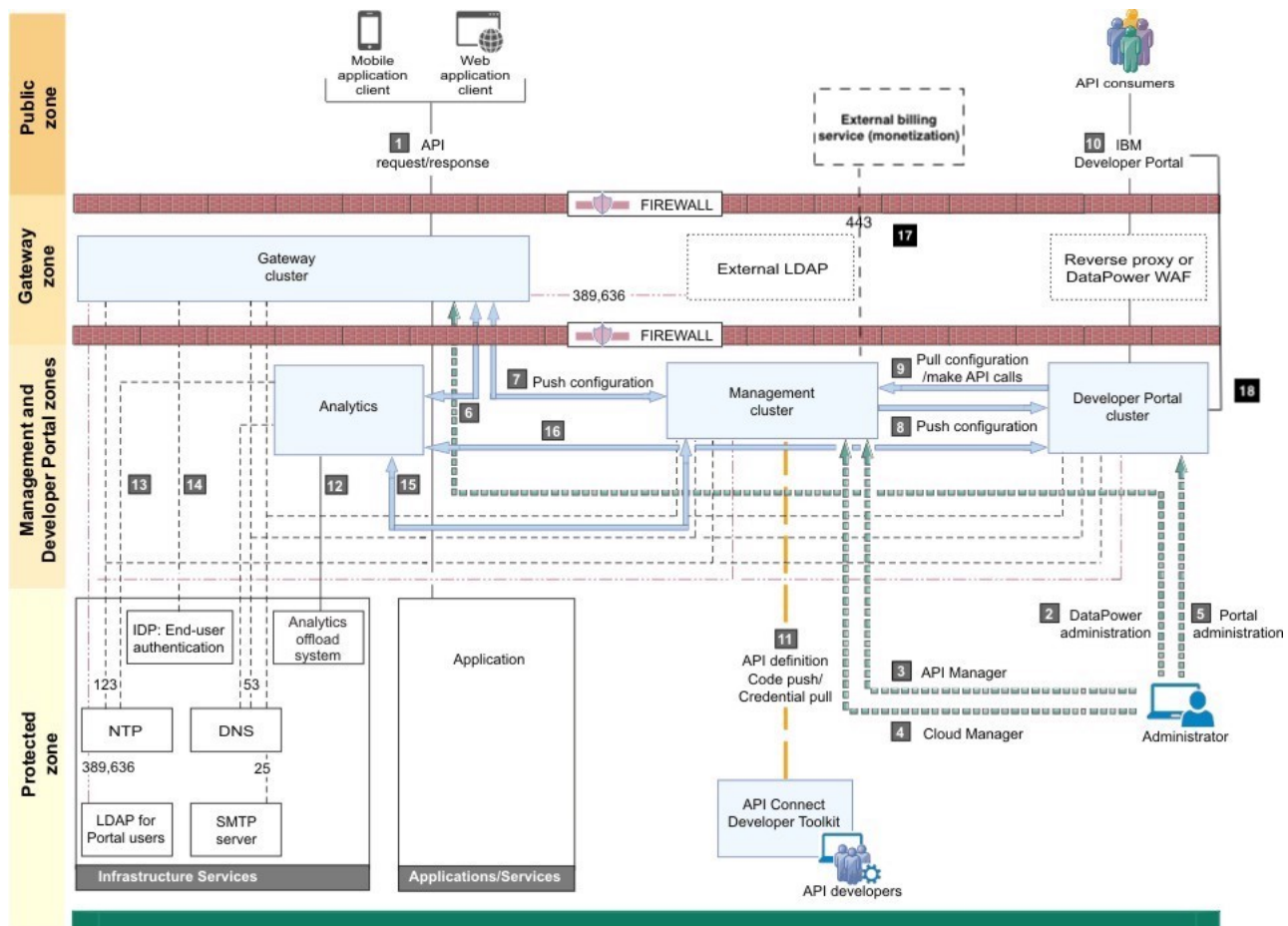


Table 1. Key for the network diagram example.
The following table lists the port numbers with a usage description.

	Usage description	Default port number
1	API request/response – Users invoking the provided APIs.	443 HTTPS from Public zone to Gateway zone
2	DataPower® administration – Internal operators who are managing the Gateway servers.	22 SSH, 9090 HTTPS from Protected zone to Gateway zone
3	API Manager – Internal business users who are defining and monitoring APIs.	443 HTTPS from Protected zone to Management zone
4	Cloud Manager – Internal operators who are administering the Cloud.	22 SSH, 443 HTTPS from Protected zone to Management zone
5	Developer Portal administration – Internal operators who are managing the Portal servers.	22 SSH, 443 HTTPS from Protected zone to Management zone
6	Gateway servers post traffic to Analytics service.	443 HTTPS from Gateway servers to Analytics service
7	Push configuration – Management servers communicate bi-directionally with Gateway servers.	3000 and 443 HTTPS Management servers to and from Gateway servers for webhook delivery. Port 3000 is the default port for apic-gw-service, and this is the port Management uses to communicate with Gateway. Port 443 is the default port for the platform-api endpoint, which apic-gw-service uses to communicate with Management.
8	Push configuration/webhooks – Management servers push configuration and webhooks to the Developer Portal.	443 HTTPS Management servers to Developer Portal servers for webhook delivery
9	Pull configuration/make API calls – Developer Portal servers pull configuration and call REST APIs.	443 HTTPS from Developer Portal servers to Management servers within Management zone
10	Developer Portal – External developers who are accessing the Developer Portal.	443 HTTPS from Public zone to Developer Portal management zone. The reverse proxy/DataPower WAF for incoming web traffic to the Developer Portal cluster must be a transparent proxy - no modification of the portal URL, port, host name or path is allowed.
11	Push API definition to Management server. Pick up credential for microservice code push.	443 HTTPS from Protected zone to Management zone

	Usage description	Default port number
12	Analytics offload	Port will depend on type of plugin and protocol used for the offload. Some possible protocols are: HTTP, HTTPS, TCP, UDP, KAFKA
13	Analytics accesses NTP	Standard NTP
14	Analytics access DNS	Standard DNS
15	Management service queries Analytics service	443 HTTPS within Management zone
16	The Portal service invokes an API (GET) on the Analytics service to retrieve data.	443 HTTPS within Management zone
17	External billing service – Management cluster connecting to external billing service (when configured for billing). If you are using Stripe as your external billing service, you must enable connections with the Stripe API at: https://stripe.com/files/jps/jps_api.json . You can also view the Stripe IP addresses at the following URL: https://stripe.com/docs/jps .	443 HTTPS from Management zone to Public zone
18	Developer Portal cluster must be able to access its own site endpoints.	443 HTTPS from Portal zone to Public zone

Firewall port requirements on Kubernetes

The following tables lists ports that must be open in both cluster and non-clustered deployments.

Table 2. Firewall port requirements common to all subsystems

Subsystem	Ports and description
Ports that must be open on all API Connect subsystems	<p>The following ports must be open on the Management Server, Analytics, and Developer Portal subsystems, whether in a cluster or not.</p> <ul style="list-style-type: none"> • 22 (outbound) Used for SFTP backups on Management and Portal subsystems • 53 (outbound) DNS • 123 (outbound) NTP • 443 (inbound and outbound) Used by all subsystems for communication with other subsystems • 44135 (inbound and outbound)

Each subsystem uses ports in addition to the ports in [Table 2](#). See the following table.

Table 3. Additional firewall port requirements for each subsystem

Subsystem	Ports and description
Management Service	<p>Management Service uses the ports in Table 2 plus:</p> <ul style="list-style-type: none"> • 22 remote backup server port (configurable) If backups are configured to use another port, ensure that the port is open. • 161 (inbound) SNMP • LDAP server port (if using LDAP user registry), typically 389 (outbound) • 3007 (outbound) LDAP <p>The Automated test behavior application must be able to invoke (outbound) any port that a test system might present an API on. It can be bound to a particular port (if it is the same in all environments) or to a range of ports (to cover all of the ports it might invoke).</p>
Developer Portal	<p>The Developer Portal uses the ports listed in Table 2, plus:</p> <ul style="list-style-type: none"> • 22 - A remote backup server port (configurable) If backups are configured to use another port, ensure that the port is open. See Backing up and restoring the Developer Portal in a Kubernetes environment.
Analytics	<p>The Analytics subsystem uses the ports listed in Table 2.</p> <p>Analytics port usage considerations:</p> <ul style="list-style-type: none"> • 161 for SNMP is required for both non-clustered and clustered deployments • In a non-clustered deployment, no additional ports are required. • Analytics supports an optional configuration for offload of data. This configuration might require additional outbound ports to be open.
Gateway Server	<p>The Gateway Server uses these ports in both non-clustered and clustered deployments:</p> <ul style="list-style-type: none"> • 161 (inbound and outbound) SNMP • 162 (outbound) SNMP traps • 3000 (inbound) Gateway Service local port (configurable) • 5550 (inbound) XML management port (configurable) • 5554 (inbound) REST management port (if enabled; configurable) • 9022 (inbound) Gateway SSH (if enabled; configurable) • 9090 (inbound) Web GUI console (if enabled; configurable) • 9443 (inbound) Gateway local port (configurable)

Communications inside the Gateway cluster

There are a number of important points to note regarding the communications within the Gateway cluster.

- We advise that you use the same port for all Gateway servers within a cluster.
- Gateway servers communicate with each other to synchronize invocation counts.
- All Gateway servers in a Gateway cluster must be able to reach all of the other Gateway servers in the same Gateway cluster.
- Gateway servers in a Gateway cluster do not directly communicate with Gateway servers in a different Gateway cluster.

- All Gateway servers must be able to reach the management subsystem platform API endpoint, which was configured during the installation of your API Connect environment.

Ethernet interface usage

To separate network traffic, you can use two or more Ethernet interfaces on the DataPower appliance on which a Gateway server is installed. For example, you can use one interface for internal IBM API Connect communications, and another for processing incoming API calls.

Port requirement for Stripe billing service

All Management servers and Developer Portal servers must be able to communicate HTTPS content with the external billing service when billing is configured. If you are using Stripe as your external billing service, you must enable HTTPS communication on port 443 with the Stripe API: https://stripe.com/files/ips/ips_api.json.

Note: API Connect does not support using webhooks with Stripe. You can see a list of the Stripe IP addresses at: <https://stripe.com/docs/ips>.

Related concepts

- [Installing and maintaining IBM API Connect](#)

Related reference

- [IBM API Connect Version 10 software product compatibility requirements](#)

Planning your analytics deployment

Plan your IBM® API Connect analytics deployment by reviewing options for topology, data modifications, and storage.

Planning your Analytics topology is an important step in ensuring a successful and scalable solution. After you plan your overall API Connect topology, plan the analytics subsystem deployment by:

- [Planning the analytics profile, storage class, and storage type](#)
Begin planning your IBM API Connect analytics deployment by choosing a deployment profile, a Kubernetes storage class, and an OpenSearch storage type.
- [Planning to modify your incoming analytics data](#)
Review how you can optionally customize the analytics data in your IBM API Connect analytics deployment before storing it.
- [Estimating internal storage space](#)
If you plan to store data locally in your IBM API Connect analytics deployment, estimate disk space requirements.

Planning the analytics profile, storage class, and storage type

Begin planning your IBM® API Connect analytics deployment by choosing a deployment profile, a Kubernetes storage class, and an OpenSearch storage type.

Deployment profile

Decide which deployment **profile** you want to use, the available profiles are listed on [Analytics component deployment profile limits](#).

Kubernetes storage class

Before you install analytics, you must choose a storage class. **Ceph**, **Block**, and **Local Volume** are all supported; however, **Local Volume** is most suitable for analytics. The analytics subsystem provides internal data replication and HA support, so the additional HA capabilities that **Ceph** or **Block** storage might give you are not needed. In addition, analytics requires a high throughput of disk I/O operations to successfully handle the analytics data load from the gateway. **Local Volume** is the best-performing option in this area.

Note: **GlusterFS** storage is not supported for analytics. If you use this option, you might encounter severe performance degradation, and possibly loss of data.

OpenSearch storage type

Note: Storage type selection is available on three replica deployments only. On a one replica deployment, only shared storage is available. API Connect provides two OpenSearch storage types:

- **Shared storage**
A single pod provides data storage and OpenSearch cluster management. Shared storage is the default option, and the only option on a one replica deployment.
- **Dedicated storage**
OpenSearch data storage and cluster management functions are split, and run in separate pods on each worker node:
 - **storage-os-master** - Runs an OpenSearch "manager eligible" node that manages the OpenSearch cluster, but does not store any API event data.
 - **storage** - Runs an OpenSearch "data" node that stores API event data.

Dedicated storage facilitates horizontal scaling to support greater analytics data storage and throughput. Add worker nodes and increase the number of **storage** pod replicas in order to scale horizontally (you do not need to increase the number of **storage-os-master** replicas). Dedicated storage also provides greater stability and allows some OpenSearch configuration changes to be made without downtime.

You can change storage type after installation, see [Changing storage type between shared and dedicated](#).

Planning your topology

Determine the topology of your IBM® API Connect analytics deployment according to your needs.

Default topology

The default topology for analytics is useful for many organizations. It provides you with business analytics and metrics in the API Manager UI, and consumer analytics in portal UI. It is suitable for production environments and is easily scalable.

The default topology contains the following microservices:

- **osinit**: OpenSearch initialization job.
- **director**: NodeJS microservice handling inbound communication, service registration, proxies calls to storage and ingestion services.
- **ingestion**: Logstash-based data pipeline for receiving and routing data to internal or external storage.
- **mtls-gw**: OpenResty-based service for routing ingress traffic to director or ingestion.
- **storage**: OpenSearch-based service for writing, reading, and storing the data.
- **storage-os-master**: If dedicated storage type is selected, **storage-os-master** manages the OpenSearch cluster, allowing the **storage** service to be dedicated to analytics data storage.

For the complete list of available microservices, see [Analytics pods and PVCs](#).

Offload topology

The offload topology disables local storage of API Connect analytics data. Instead, the data is sent to a third-party system, and it is not viewable in the API Connect UIs.

The offload topology contains the following microservices:

- **director**: NodeJS microservice handling inbound communication, service registration, proxies calls to storage and ingestion services.
- **ingestion**: Logstash-based data pipeline for receiving and routing data to internal or external storage.
- **mtls-gw**: OpenResty-based service for routing ingress traffic to director or ingestion.

For information about configuring offload settings, see [Planning to offload data to a third-party system](#).

Internal and external storage topology

You can also have both internal storage and offload enabled, so that you can view your analytics data in the API Connect UIs, and store the data externally. This topology has the same set of microservices as the default topology.

For more information about disabling internal storage, see [Planning to disable internal storage](#).

Analytics pods and PVCs

See the complete list of pods and physical volume claims that comprise an API Connect analytics deployment.

Table 1 provides the name and a brief description of each microservice.

Table 1. Analytics microservices

Name	Description
osinit	OpenSearch initialization job.
director	NodeJS microservice handling inbound communication, service registration, proxies calls to storage and ingestion.
ingestion	Logstash-based data pipeline for receiving and routing data to internal or external storage.
mtls-gw	OpenResty-based service for routing ingress traffic to client or ingestion.
storage	If the shared storage type is selected, the storage pods contain both your analytics data and the OpenSearch cluster management processes. If the dedicated storage type is selected, the storage pods contain just the analytics data.
storage-os-master	If the dedicated storage topology is selected, the storage-os-master pods are created and manage the OpenSearch cluster in your deployment.

Table 2. Analytics PVCs

Name	Description
data-analytics-storage- <integer>	A PVC is created for each analytics storage pod. <integer> is the replica number, starting from 0.
data-analytics-storage-os- master- <integer>	If dedicated storage type is selected, then the storage-os-master PVCs are created for the storage-os-master pods.
data-analytics-ingestion- <integer>	If the persistent queue feature is enabled, a PVC is created for each ingestion pod. <integer> is the replica number, starting from 0. See Configuring the persistent queue .

Planning to offload data to a third-party system

Considerations for offloading data from your IBM® API Connect analytics deployment to a third-party system.

The analytics event data that sent from the gateway to analytics can be offloaded to a third-party service in real-time. The third-party service can provide enhanced monitoring, alerting, storage, and analysis, in addition to what the built-in analytics subsystem provides. Multiple target systems are supported, and the data can be

modified before offloading.

APIC supports offloading analytics data to the following third-party systems:

- HTTPs endpoints
- OpenSearch
- Elasticsearch
- Apache Kafka
- Syslog servers

For instructions on configuring your offload settings, see [Kubernetes - Offloading data to a third-party system](#), [OpenShift - Offloading data to a third-party system](#).

If you do not need internal storage and want to use only the third-party system for working with your analytics data, consider disabling internal storage entirely as explained in [Planning to disable internal storage](#).

Planning to disable internal storage

Review considerations for disabling local storage and offloading all data in your IBM® API Connect Analytics deployment.

The Analytics subsystem provides a complete solution for routing, storing, and viewing the analytics data. However, you might not need the internal storage and viewing solution, especially if you are primarily using a third-party system for data offloading. In this scenario, you can greatly reduce your CPU and memory costs by disabling the internal storage and viewing components.

With this approach, data that is offloaded is not viewable in the API Manager and Developer Portal UIs.

When you disable internal storage for analytics data, the following microservices are disabled:

- `osinit`
- `storage`

Important: The disablement of internal storage must be set in the analytics CR yaml file before installation. It is not possible to disable or re-enable internal storage after installation.

For the steps to disable internal storage, see [Kubernetes - Disabling internal storage](#), [OpenShift - Disabling internal storage](#).

Planning to modify your incoming analytics data

Review how you can optionally customize the analytics data in your IBM® API Connect analytics deployment before storing it.

You can create filters to customize the analytics API event data before it is sent to internal storage or offloaded to a third-party system. Filters allow you to add new fields, modify existing fields, or remove fields from the data. You can do this globally for the entire analytics subsystem, or you can do it conditionally for a specific API. By default, filters are defined at the global level, so unless you include conditionals, changes are applied to every piece of data flowing through the pipeline.

For information on adding filters to the analytics CR to modify your data, see [Kubernetes - Modify incoming analytics data](#), [OpenShift - Modify incoming analytics data](#).

What fields can you modify?

The fields that are available for you to interact with are described in [API event record fields](#). Each API defined in the Management subsystem and published to your Gateway has its own success and error log policies. The log policy settings determine what data is available for customization. Be sure to verify the existence of a field before trying to modify it; otherwise your pipeline might fail.

Modifying fields is a complex operation and can cause problems with your data. It is highly recommended that you remove fields entirely (instead of modifying them) if you need to sanitize data. If you want to change the format of an existing field, you should create a new, unique field.

Important: Do not change the data type of existing fields or modify fields critical to the analytics subsystem's operations and access control. Doing so can cause problems with retrieving or storing data. The following fields are restricted (note that there might be other fields that are not included in this list):

- `org_id`
- `catalog_id`
- `space_id`
- `developer_org_id`
- `datetime`
- `@timestamp`

The service for the data pipeline is based on Elastic Logstash, which provides [a set of predefined filter plugins](#) that you can use for modifying data. However, the plugins are third-party software that IBM does not control, so IBM cannot guarantee support for them.

Estimating internal storage space

If you plan to store data locally in your IBM® API Connect analytics deployment, estimate disk space requirements.

This information applies only to data that is stored in API Connect analytics and not to data that is offloaded to a third-party system. The formula and guidelines are based on known information about how the analytics service stores data, and cannot be directly applied to any other storage system.

Use the following guidelines to calculate a rough estimate of the amount disk space you need for storing stateful data in the API Connect analytics microservices.

Data storage: calculate how much data you want to store

The amount of disk space needed for storing analytics data is determined by the following factors:

Number of copies of the analytics data

Each copy of the analytics data must live on a separate node. The number of nodes in your deployment (based on your chosen profile in [Planning the analytics profile, storage class, and storage type](#)) determines the number of copies of analytics data that is stored in your deployment. With the three replica deployment profile, analytics data is replicated to three nodes (two replicas and one primary copy of the data) for storage. In the one replica profile, you only need to store one copy of the data.

Number of copies of data:

```
Copies of data = [1 | 3]
```

Number of days that data is retained

By default, analytics data is retained for 90 days, but you can [modify the retention setting](#) as needed. Make sure you know how long you want to store the data before attempting to calculate required disk space.

Number of days that data is retained:

```
Data retention = [90 days | preferred length]
```

Amount of each type of data stored

The required storage space for each type of logging is highly dependent on your APIs and usage. For each API, you can [configure activity logging](#) for the API activity, header, and payload. You can also [customize the data](#) to add, redact, or remove fields, which also impacts the amount of data that you store.

To estimate storage needs, calculate the average size of each type of log. When calculating your estimates, remember that the header logging size is the sum of activity logging size and the average size of your headers. The payload logging size is the sum of the header logging size and the average size of your payloads. Typically the average size of an activity logged event is 600-1000 bytes depending on the uniqueness and complexity of your analytics data. This number is highly dependent on your APIs and your implementations. For a rough estimate, you can use an average of 800 bytes per activity logged event.

If you choose to add fields, calculate the average size of the new fields as well, and add that number to all types of log policies. If you choose to remove fields, you should not subtract this size from the log policies unless you are also removing headers and/or payloads.

Amount of each type of data that is stored:

```
Activity log bytes per call = [600-1000 bytes]
Header log bytes per call = Activity log bytes per call + Average size of headers
Payload log bytes per call = Header log bytes per call + Average size of payloads
```

Percentage of each type of data

Estimate the percentage of each type of log (activity, header, payload) for all API calls.

If you follow best practices of using only activity logging for production environments and using only payload logging for test environments, this number is easy to determine. If you use different log policies per API, and they depend on "success" or "error" factors, the percentage is more difficult to determine. Typically, if you do not use an all-or-nothing method for logging, error rates range from 3% to 25% with subsequent payload logging in test and production environments. However, this is entirely dependent on your use case and your APIs.

Percentage of each type of data that is stored:

```
% of Activity log = [0, 100 or other estimate]
% of Header log = [0, 100 or other estimate]
% of Payload log = [0, 100 or other estimate]
```

Estimated number of API calls per month

When planning your API Connect deployment, this number is helpful. When estimating analytics storage, this number is vital because it is directly correlated to how much storage you need for your deployment.

If you do not know the number of calls per month, but you do know the number of calls per second, use the following formula to convert it to calls per month:

```
Calls per month = Calls per second * 86400 seconds per day * 30 days per month
```

Number of API calls per month:

```
Calls per month = [any estimate]
```

Calculating your disk space requirement

Estimate the disk space requirement for each storage node by completing following calculations.

Formula

```
Bytes per call = (% of Activity Log * Activity Log bytes per call) + (% of Header Log * Header Log bytes per call) + (% of
Payload Log * Payload Log bytes per call)
Calls per retention period = Calls per month * (Number of days retained / 30 days per month)
Storage for API calls = Calls per retention period * Bytes per call * Copies of data
Total storage = Storage for API calls + Overhead
Storage per node = Total storage / Nodes
```

Details

1. Bytes per call:

Estimate the number of bytes that are logged for a single copy of each API call. This can be calculated from the prerequisites of the percentage of each data type and the amount of each data type stored.

```
Bytes per call = (% of Activity Log * Activity Log bytes per call) + (% of Header Log * Header Log bytes per call) +
(% of Payload Log * Payload Log bytes per call)
```

2. Calls per retention period:

Calculate the anticipated number of API calls per retention period. This can be calculated from the prerequisites of the estimated calls per month and your desired data retention.

Calls per retention period = Calls per month * (Number of days retained / 30 days per month)

3. Storage for API Calls:

Calculate the storage needed for all copies of your api calls for your retention period. This can be calculated from steps 1 and 2, as well as the prerequisite of your total copies of your data.

Storage for API calls = Calls per retention period * Bytes per call * Copies of data

4. Total Storage:

Calculate the total storage you need by adding buffer space to the value from step 3. The analytics service requires some overhead space to complete its operations; for example, to contain system data and temporary debug header or payload logging. In addition, allowing extra space provides a buffer in case you underestimated the number of calls, or experience an unexpected increase.

There is no specific value for the buffer because it's based on your own situation. One approach is to use a value that rounds the Storage for API calls result from step 3 to the next whole number. Make sure the rounding leaves you with a comfortable amount of additional space. For example, if the result from step 3 is 380 GB, then adding 20 GB to reach 400 GB is probably not sufficient and you should consider rounding to a larger value such as 500 GB.

Total storage = Storage for API calls + Buffer

5. Storage per node:

Calculate the total storage amount required per storage node. The number of storage nodes is dependent on your deployment profile. For the one replica profile, use 1. For the three replica profile, it defaults to 3. If you manually scaled the storage microservices to be greater than 3, use your actual values.

Storage per node = Total storage / Nodes

Remember: This result is only an estimate. You should monitor the use of space over time and adjust storage as needed.

Save this information for when you are updating the analytics CR for your installation. [Kubernetes - Creating the analytics CR](#), [OpenShift - Analytics CR settings](#)

Example

Deployment information:

Copies of data = 3
Data retention = 90 days
Activity log bytes per call = 850 bytes
Header log bytes per call = 15k bytes
Payload log bytes per call = 30.5k bytes
% of Activity log = 100%
% of Header log = 0%
% of Payload log = 0%
Calls per month = 64 million

Formula:

Bytes per call = (% of Activity Log * Activity Log bytes per call) + (% of Header Log * Header Log bytes per call) + (% of Payload Log * Payload Log bytes per call)
Calls per retention period = Calls per month * (Number of days retained / 30 days per month)
Storage for API calls = Calls per retention period * Bytes per call * Copies of data
Total storage = Storage for API calls + Overhead
Storage per node = Total storage / Nodes

Details:

1. Bytes per call = 850 bytes

(100% of Activity Log * 850 bytes) + (0% of Header Log * 15k bytes) + (0% of Payload Log * 30.5 bytes)

2. Calls per retention period = 192 million calls

64 million calls per month * (90 days retained / 30 days per month)

3. Storage for API calls = 489.6 GB

192 million calls per period * 850 bytes per call * 3 copies of data

4. Total storage = 600 GB

489.6GB storage + Buffer

Since rounding to 500 GB only provides 10.4 GB of extra space, consider rounding to 600 GB instead.

5. Storage per node = 200 GB

600GB Total storage / 3 nodes

In this example, the estimated disk needed on each node for storage microservices is 200GB.

Two data center deployment strategy on Kubernetes and OpenShift

An overview of the two data center disaster recovery deployment strategy in API Connect.

Key points of the two data center disaster recovery (DR) solution

- Two data center DR is an active/warm-standby deployment for the API Manager and Developer Portal services, and must use manual failover.
- Two DataPower® Gateway subsystems must be deployed to provide high availability for the gateway service. However, this scenario doesn't provide high availability for the analytics service.
- If high availability is required for the analytics service, two analytics subsystems must be configured, one per gateway subsystem, but with this configuration Developer Portal analytics isn't possible.

- Data consistency is prioritized over data availability.
- Each data center must be setup within its own Kubernetes or OpenShift cluster.
- Latency between the two data centers must be less than 80 ms.
- Replication of the API Manager is asynchronous, so it is possible that the most recent updates do not transfer to the warm-standby data center if there is an active data center failure.
- Replication of the Developer Portal is synchronous, and therefore the latency is limited to 80 ms or less.
- The API Manager and the Developer Portal services in the two data centers must use the same deployment profile.
- The deployment in each data center is effectively an instance of the same API Connect deployment - therefore, the endpoints, certificates, and Kubernetes secrets must all be the same.
- It is not possible to use the Automated API behavior testing application ([Installing the Automated API behavior testing application](#)) in a two data center disaster recovery configuration.

Deployment architecture

A two data center deployment model is optimized for data consistency ahead of data availability, and must use manual failover when a fault occurs. For high availability of Management and Portal subsystems ensure that you use a three replica deployment profile. For more information on deployment profiles, see: [Planning your deployment topology](#).

For a single Kubernetes cluster to span multiple data centers the network latency must be no more than a few milliseconds, typically less than 10 ms. This low latency is often unachievable between geographically separated data centers. For this reason, the two data center disaster recovery solution requires that each data center is set up with its own Kubernetes cluster. The Management and Portal subsystem databases are continually replicated from the active datacenter to the warm-standby datacenter, this requires a network latency between the two data centers of less than 80 ms.

To achieve high availability for the DataPower Gateway, you must deploy two gateway subsystems. One subsystem in the active data center, and a separate subsystem in the warm-standby data center. Publish all Products and APIs to both gateway subsystems. The gateway subsystems are independent, and so are insulated if an issue occurs in one of them. A global dynamic router can then be used to route traffic to one gateway subsystem or the other. If high availability is also required for the analytics service, two analytics subsystems must be configured, one per gateway subsystem, but with this configuration Developer Portal analytics isn't possible.

There can be multiple Developer Portal services in an API Connect deployment (although still only one Developer Portal site per Catalog).

The deployment in each data center is effectively an instance of the same API Connect deployment – as the database is replicated, all of the configuration is also shared. Therefore, the endpoints, certificates, and Kubernetes secrets all need to be the same.

Dynamic routing

Note: A dynamic routing device, such as a load balancer, is required to route traffic to either data center. However, neither this device nor its configuration is part of the API Connect offering. Contact IBM Services if you require assistance with configuring a dynamic router.

The dynamic router configuration must handle the traffic between the subsystems and between the data centers. For example, the consumer API calls from the Developer Portal to API Manager must transfer through a dynamic router so that the Developer Portal can use a single endpoint regardless of which data center API Manager is active in. The same is needed for calls to the Platform and Admin APIs from the other subsystems, as well as for incoming UI traffic for the Developer Portal UI, Cloud Manager UI, and API Manager UI.

The dynamic router must support SSL passthrough, so that it routes the Mutual TLS (mTLS) connections between API Manager and the Developer Portal, and between API Manager and the DataPower Gateway. The router should not do TLS termination, it should do layer 4 based routing by using SNI.

Note: From v10.0.5.3, it is possible to disable mTLS and use JWT instead, which allows the load-balancers to do TLS termination. For more information, see [Enable JWT security instead of mTLS](#).

Service status

When a failure occurs, it is common practice to display an interstitial system outage web page. The dynamic router can be configured to display this web page when there is a failure, while the manual failover to the warm-standby data center is taking place.

Deployment-profiles

Both one replica and three replica deployment profiles can be used with two data center DR, but they must be the same at each data center. For more information about the deployment-profiles, see [Planning your deployment topology](#), on Kubernetes, and [Requirements for initial deployment on VMware](#).

For more information about a two data center DR deployment, see the following topics:

Kubernetes and OpenShift

- [Installing a two data center deployment on Kubernetes](#)
 - [Converting a single data center to a two data center deployment on Kubernetes](#)
- [Maintaining a two data center deployment](#) (including information about normal operation data flows)
 - [How to failover API Connect from the active to the warm-standby data center](#)
 - [Recovering from a failover of a two data center deployment](#)
 - [Backup and restore requirements for a two data center deployment](#)
 - [Upgrading a two data center deployment on Kubernetes and OpenShift](#)
 - [Removing a two data center deployment](#)

Key Concepts: Cert-manager, Issuers, and Secrets

By default API Connect uses an open source product that is called [cert-manager](#) to handle the issuing and renewal of the certificates that are used by API Connect. The cert-manager has its own Kubernetes pods and runs in its own namespace. The cert-manager adds some additional resources to the Kubernetes environment. The API Connect administrator needs to be familiar with these additional resources:

Issuers

Issuers play the part of certificate authorities (CAs) in the local environment and issue the certificates. Issuers reside in the API Connect namespace. There are two types of Issuers:

- Self-Signed - This issuer provides the self-signed certificates, for example a root CA.

- CA - This issuer can be either the root certificate authority or an intermediate certificate authority.

Here is an example of the issuers that can be found on the management subsystem:

```
kubectl get issuers -n <management namespace>
NAME                READY   AGE
abc-management-ca   True    96d
selfsigning-issuer  True    96d
```

- **selfsigning-issuer** This is a SelfSigned Issuer that provides the root certificate for all default certificates that are used in API Connect.
- **abc-management-ca** This is a CA Issuer that signs all the end-entity certificates that are used for communication between the management subsystem pods. It is in effect an intermediate CA between the root (**selfsigning-issuer**) and the end-entity certificates.

Certificates

A certificate is a Kubernetes resource that contains a pointer to a Kubernetes secret. The Kubernetes secret contains the actual TLS certificate. Certificate objects contain other information such as the issuer, expiry date, and status. Run `kubectl get certificate` to see the certificates on an API Connect subsystem:

```
kubectl get certificate -n <management namespace>
NAME                READY   SECRET                AGE
abc-management-ca   True    abc-management-ca     96d
abc-management-client True    abc-management-client 96d
abc-management-natscluster-mgmt True    abc-management-natscluster-mgmt 96d
abc-management-server True    abc-management-server 96d
...
```

The first certificate in the list **abc-management-ca** is the intermediate CA certificate that is used for signing the other certificates that are shown in the output. If you describe this certificate you can see that the **selfsigning-issuer** is its issuer:

```
kubectl describe certificate abc-management-ca -n <management namespace>
...
Spec:
...
  Issuer Ref:
    Kind: Issuer
    Name: selfsigning-issuer
```

Describing the other management subsystem certificates, observe that the **abc-management-ca** certificate is the issuer:

```
kubectl describe certificates abc-management-client -n <management namespace>
...
Spec:
...
  Issuer Ref:
    Kind: Issuer
    Name: abc-management-ca
```

Key points:

- The certificates with '-ca' on the end are all CA Certificates, and they sign all the other subsystem certificates whose names do not end with '-ca'.
- The CA certificates are also known as Issuer certificates.
- Certificates that CA certificates sign are known as end-entity certificates.
- When a CA certificate is updated, all end-entity certificates that it signs must also be updated.

For a complete list of the certificates in an API Connect installation, see [API Connect certificates reference](#).

Secrets

Secrets are the Kubernetes resource for storing credentials. In API Connect, secrets are where the TLS certificates are stored. Every Kubernetes certificate object has a corresponding secret of the same name. The Kubernetes certificate contains a property that refers to its corresponding secret:

```
kubectl describe certificates abc-management-client -n <management namespace>
...
Spec:
...
  Secret Name: abc-management-client
```

The default output of `kubectl get certificate` includes a column that is labeled **SECRET**, which shows the secret name.

```
kubectl get certificate -n <management namespace>
NAME                READY   SECRET                AGE
abc-management-ca   True    abc-management-ca     96d
abc-management-client True    abc-management-client 96d
abc-management-natscluster-mgmt True    abc-management-natscluster-mgmt 96d
abc-management-server True    abc-management-server 96d
...
```

The `kubectl describe` output of a secret shows that it contains the TLS certificates:

```
kubectl describe secret abc-management-client -n <management namespace>
...
Name: abc-management-client
Namespace: default
Labels: app.kubernetes.io/instance=abc-management
        app.kubernetes.io/managed-by=ibm-apiconnect
        app.kubernetes.io/name=abc-management-client
Annotations: cert-manager.io/alt-names:
             cert-manager.io/certificate-name: abc-management-client
             cert-manager.io/common-name: abc-management-client
             cert-manager.io/ip-sans:
             cert-manager.io/issuer-group:
```



```
cert-manager.io/issuer-kind: Issuer
cert-manager.io/issuer-name: abc-management-ca
cert-manager.io/uri-sans:
```

Type: kubernetes.io/tls

Data

====

```
ca.crt: 1107 bytes
tls.crt: 1139 bytes
tls.key: 1675 bytes
```

The full TLS certificate strings can be seen with:

```
kubecttl get -o yaml secret <secret name> -n <namespace>
```

For more details about cert-manager, and a full list of all the certificates that are created and used in an API Connect deployment, see [API Connect certificate reference](#).

Certificates in a Kubernetes environment

Use of cert-manager is recommended for managing certificates in a Kubernetes environment.

When deploying on Kubernetes, choose one of the following methods for creating internal certificates:

- Do not specify any internal certificates. In this case, the operator generates all of the internal certificates without using a certificate manager. The CR definition to enable this option is:

```
microServiceSecurity: custom
```

- Specify some, or all, of the internal certificates yourself without using a certificate manager. In this case, any certificates that you do not create are created by the operator. You retain control over the certificates that you create. The CR definition to enable this option is:

```
microServiceSecurity: custom
```

- Configure cert-manager to generate the internal certificates. In this case, cert-manager creates and manages the certificates. The CR definition to enable this option is:

```
microServiceSecurity: certManager
```

To use cert-manager for generating the certificates, download and install cert-manager v1.9.1 from <https://github.com/cert-manager/cert-manager/releases/tag/v1.9.1> and create an Issuer CR in the same namespace where you will install API Connect; for example:

```
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
spec:
  selfSigned: {}
```

In the CR for the API Connect Management subsystem, specify a `certManagerIssuer`. The `certManagerIssuer` can be configured to be a self-signing issuer, as in the following example:

```
microServiceSecurity: certManager
certManagerIssuer:
  name: selfsigning-issuer
  kind: Issuer
```

API Connect supports any other kind of Issuer supported by cert-manager, in addition to the self signed type.

In all the CRs, certs are expected at the level of any endpoint. For example, for the cloud admin endpoint of the Management subsystem you could have:

```
cloudManagerEndpoint:
  annotations:
    cert-manager.io/issuer: ingress-issuer
  hosts:
  - name: admin.example.com
    secretName: cm-endpoint
```

The previous entry lets cert-manager automatically generate a `cm-endpoint` TLS secret using the `ingress-issuer`. One way to define the `ingress-issuer` is as follows:

```
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ingress-ca
spec:
  secretName: ingress-ca
  commonName: "ingress-ca"
  usages:
  - digital signature
```

```

- key encipherment
- cert sign
isCA: true
duration: 87600h # 10 years
renewBefore: 720h # 30 days
privateKey:
  rotationPolicy: Always
issuerRef:
  name: selfsigning-issuer
  kind: Issuer
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
spec:
  ca:
    secretName: ingress-ca

```

Another way to configure the cloud admin endpoint would be with:

```

cloudManagerEndpoint:
  hosts:
    - name: admin.example.com
      secretName: my-cm-endpoint

```

Where `my-cm-endpoint` is the name of a secret that you previously created, and is present in the same namespace as the subsystem, and contains TLS certificate files for termination of the endpoint.

Creating secrets and generating certificates

There is a number of "secrets" for which you can provide a name in the CRs:

- Portal admin secret.

```

portal:
  admin:
    secretName:

```

- Analytics ingestion secret.

```

analytics:
  ingestion:
    secretName: analytics-ingestion-client

```

- APIC Gateway Service TLS secret. Template entry from `gateway_cr_v6.yaml` (in the file name, "v6" refers to the API Gateway service):

```

apicGatewayServiceTLS:
  secretName:

```

- APIC Gateway Peering TLS secret. Template entry from `gateway_cr_v6.yaml` (in the file name, "v6" refers to the API Gateway service):

```

apicGatewayPeeringTLS:
  secretName:

```

You can create manually create secrets that contain the `tls.crt`, `tls.key`, and `tls.ca` files for a certificate. You can then refer to them by name in their subsystem CR. However, a simpler way to have those secrets created is by using `cert-manager`. The following example shows how to generate those certificates with `cert-manager`:

```

---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ingress-ca
spec:
  secretName: ingress-ca
  commonName: "ingress-ca"
  usages:
    - digital signature
    - key encipherment
    - cert sign
  isCA: true
  duration: 87600h # 10 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  issuerRef:
    name: selfsigning-issuer
    kind: Issuer
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
spec:
  ca:

```

```

    secretName: ingress-ca
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: portal-admin-client
spec:
  commonName: portal-admin-client
  secretName: portal-admin-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: analytics-ingestion-client
spec:
  commonName: analytics-ingestion-client
  secretName: analytics-ingestion-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-service
spec:
  commonName: gateway-service
  secretName: gateway-service
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-peering
spec:
  commonName: gateway-peering
  secretName: gateway-peering
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always

```

So with the previous example, in the subsystem CRs (for Management and Gateway) the following values could be set:

- portal:
 - admin:
 - secretName: portal-admin-client
- analytics:
 - ingestion:
 - secretName: analytics-ingestion-client
- apicGatewayServiceTLS:
 - secretName: gateway-service
- apicGatewayPeeringTLS:
 - secretName: gateway-peering

Also, the Client certificate should be signed by same CA as the one for the endpoint certificates, and the server side should be configured with the expected Subject DN for the client. For example:

- The `portalAdminSecret` certificate and the certificate used to terminate `portalAdminEndpoint` must have the same CA (in the example the CA is `ingress-ca` for both). Also, the `adminClientSubjectDN` in the Portal subsystem CR should be set to the Subject DN of `portal.admin.secretName`. For example:
`adminClientSubjectDN: CN=portal-admin-client,O=cert-manager`
- Similarly, the `analyticsIngestionSecret` and the Analytics ingestion endpoint certificate should share the same CA. The `ingestion.clientSubjectDN` should be set to the Subject DN of the `analytics.ingestion.secretName` certificate.

```
ingestion:
  endpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: ai.example.com
        secretName: analytics-ai-endpoint
    clientSubjectDN: CN=analytics-ingestion-client,O=cert-manager
```

Important: When installing more than one Analytics subsystem in the same Kubernetes namespace, all secret names must be unique.

Creating TLS secret using openssl

Use of the cert-manager is recommended for managing certificates. However, you can choose not to use cert-manager, and instead create certificates manually. Requirements for manually created certificates:

- Extended Key Usage (EKU), either `serverAuth` or `clientAuth` depending upon the type of certificate. Certificates of type *Server* must have an Extended Key Usage with `serverAuth` purpose. Certificates of type *Client* must have an Extended Key Usage with `clientAuth` purpose.
- Subject Alternative Name (SAN) for the required hosts
- If a certificate is signed by an internal or custom CA, include the full chain in the end certificate. If you omit the full chain, then a user who uses openssl to access the endpoint will see the following error: `error:num=20:unable to get local issuer certificate`
 The following example shows the full chain for an end certificate (signed by a custom CA):

```
-----BEGIN CERTIFICATE -----
Cert contents for end-cert
-----END CERTIFICATE -----
-----BEGIN CERTIFICATE -----
Cert contents for Intermediate-CA
-----END CERTIFICATE -----
-----BEGIN CERTIFICATE -----
Cert contents for Root CA
-----END CERTIFICATE -----
```

The following steps can be used as an alternative to using cert-manager. If you choose to create your certificates this way, you lose some of the management features of cert-manager. For example, you must manually update your certificates when their expiration date gets near.

1. Generate the custom certificate with the appropriate EKU and SAN. You will need to obtain the private key, public certificate, and CA certificates in non-password-protected PEM format for the custom certificate. Following is an example for how to generate a certificate (`platform-api-example`) with an EKU `serverAuth` and SAN using openssl:

```
openssl x509 -req -days 360 -in platform-api-example.csr -CA root-ca.pem -CAkey root-ca-key.pem -CAcreateserial -out platform-api-cert -sha256
-extfile <(cat /etc/ssl/openssl.cnf <(printf
"\n[SAN]\nsubjectAltName=DNS:fqdn.myserver.com\nnextendedKeyUsage=serverAuth"))
-extensions SAN
```

where

- `DNS:fqdn.myserver.com` is the fully qualified domain name of the endpoint the certificate applies to.
- `platform-api-example.csr` is the file name for the certificate signing request

Following is an example for how to generate a certificate (`portal-client`) with an EKU `clientAuth` and SAN using openssl:

```
openssl x509 -req -days 360 -in portal-client-example.csr -CA root-ca.pem -CAkey root-ca-key.pem -CAcreateserial -out portal-client-cert
-sha256 -extfile <(cat /etc/ssl/openssl.cnf <(printf "\n[SAN]\nkeyUsage=critical,
digitalSignature,keyEncipherment\nnextendedKeyUsage = clientAuth\nbasicConstraints=critical,
CA:FALSE\nsubjectKeyIdentifier=hash\n")) -extensions SAN
```

2. Create a Kubernetes secret containing `tls.cert`, `tls.key`, and `ca.cert`.

Setting an encryption-secret for the management database using openssl

You can optionally set the encryption-secret for the management database. If you do not set one, one is automatically generated for you.

1. The encryption-secret is a secure random bytes password used for field level encryption in the management database. You can generate 128 random bytes using the following command in openssl:

```
openssl rand -out /path/to/secret/encryption-secret.bin 128
```

2. Use `kubectl` to put the generated secret into `management-encryption-key`:

```
kubectl create secret generic management-encryption-key -n <namespace>
--from-file=/path/to/secret/encryption_secret.bin
```

where `<namespace>` is the namespace where the Management CR is going to be created)

3. Make sure that in `management_cr.yaml`, `encryptionSecret.secretName` points to the name of the secret created in the previous step.

Important: Back up the `encryption-secret.bin` file to a safe location. If this file lost it will not be possible to access the management database or its backups.

- [Custom certificates on Kubernetes](#)
You can use custom certificates in your Kubernetes deployment. You can optionally generate your own custom certificates.

Custom certificates on Kubernetes

You can use custom certificates in your Kubernetes deployment. You can optionally generate your own custom certificates.

To deploy API Connect with Custom Certificates, some additional preparation and steps are required before any subsystem CRs are applied into the Cluster.

You can provide existing custom certificates that you already own, for example DigiCert certificates, or generate new custom certificates using a package such as Cert-Manager.

- If you have your own custom certificates, see [Configuring custom certificates before installation](#).
- If you do not already have your own custom certificates, and would like to use Cert-Manager to generate custom certificates, see [Generating custom certificates](#).
- To view a list of certificates used by API Connect, see [Custom certificates reference](#)
- [Custom certificates reference](#)
API Connect supports custom certificates for each subsystem, and for internal communications.
- [Generating custom certificates](#)
You can generate custom certificates.
- [Generating custom certificates using cert-manager in a two data center deployment](#)
You can generate custom certificates in a two data center disaster recovery deployment on Kubernetes.
- [Configuring custom certificates before installation](#)
Configure custom certificates for your API Connect deployment before you begin the installation procedures.
- [Converting to custom front-end/ingress certificates after deployment](#)
Convert a front-end certificate generated by cert-manager to a custom certificate for an existing IBM® API Connect deployment.

Custom certificates reference

API Connect supports custom certificates for each subsystem, and for internal communications.

Table 1. Certificates for Management Subsystem:

Certificate	Type	Notes
<code>analytics-ingestion-client</code>	Common Subsystem Communication	Must be signed by the same CA as <code>analytics-ai-endpoint</code> Certificate of Analytics Subsystem. For a two data center deployment, both data centers must have an identical subject name. For example, both data centers subject name could be <code>CN=a7s-ingestion-client</code> , or they could both be <code>CN=a7s-ingestion-client, O=cert-manager</code> , but they must be identical.
<code>portal-admin-client</code>	Common Subsystem Communication	Must be signed by the same CA as <code>portal-admin</code> Certificate of Portal Subsystem. For a two data center deployment, both data centers must have an identical subject name. For example, both data centers subject name could be <code>CN=portal-admin-client</code> , or they could both be <code>CN=pt1-adm-client, O=cert-manager</code> , but they must be identical.
<code>gateway-client-client</code>	Common Subsystem Communication	Must be signed by the same CA as <code>gmv5-management-endpoint</code> and/or <code>gmv6-management-endpoint</code> of Gateway Subsystem
<code>cm-endpoint</code>	External Frontend/Ingress	
<code>apim-endpoint</code>	External Frontend/Ingress	
<code>api-endpoint</code>	External Frontend/Ingress	
<code>consumer-endpoint</code>	External Frontend/Ingress	
<code>hub-endpoint</code>	External Frontend/Ingress	
<code>turnstile-endpoint</code>	External Frontend/Ingress	

Note:

To generate the certificates for the endpoints used by the Automated testing behavior endpoints (`hub-endpoint` and `turnstile-endpoint`), add the following statements to the `custom-certs-external.yaml`:

```
apiVersion: cert-manager.io/v1alpha2
kind: Certificate
metadata:
  name: hub-endpoint
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "hub-endpoint"
  }
spec:
```

```

commonName: hub-endpoint
secretName: hub-endpoint
dnsNames:
- hub.example.com
issuerRef:
  name: ingress-issuer
usages:
- "server auth"
- "signing"
- "key encipherment"
duration: 17520h # 2 years
renewBefore: 720h # 30 days
privateKey:
  rotationPolicy: Always
---
apiVersion: cert-manager.io/v1alpha2
kind: Certificate
metadata:
  name: turnstile-endpoint
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "turnstile-endpoint"
  }
spec:
  commonName: turnstile-endpoint
  secretName: turnstile-endpoint
  dnsNames:
  - turnstile.example.com
  issuerRef:
    name: ingress-issuer
  usages:
  - "server auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always

```

Table 2. Certificates for Analytics Subsystem

Certificate	Type	Notes
analytics-ai-endpoint	External Frontend/Ingress	Must be signed by the same CA as <code>analytics-ingestion-client</code> Certificate of Management Subsystem

Table 3. Certificates for Gateway Subsystem

Certificate	Type	Notes
gww5-endpoint	External Frontend/Ingress	
gww5-management-endpoint	External Frontend/Ingress	Must be signed by the same CA as <code>gateway-client-client</code> Certificate of Management Subsystem
gww6-endpoint	External Frontend/Ingress	
gww6-management-endpoint	External Frontend/Ingress	Must be signed by the same CA as <code>gateway-client-client</code> Certificate of Management Subsystem

Table 4. Certificates for Portal Subsystem

Certificate	Type	Notes
portal-admin	External Frontend/Ingress	Must be signed by the same CA as <code>portal-admin-client</code> Certificate of Management Subsystem
portal-web	External Frontend/Ingress	

Table 5. Internal Certificates

Certificate	Type (CA/Server/Client)	Subsystem	Notes
caCertificate	CA	Management, Analytics, Portal	
clientCertificate	Client	Management, Analytics, Portal	
serverCertificate	Server	Management, Analytics, Portal	Portal DNS names required *.<namespace> *.<namespace>.svc *.<instance name>-server.<namespace>.svc <instance name>-server *.<instance name>-<site name>-db-all.<namespace>.svc *.<instance name>-<site name>-www-all.<namespace>.svc *.<instance name>-<site name>-db-all. <namespace>.svc.cluster.local *.<instance name>-<site name>-www-all. <namespace>.svc.cluster.local *.<namespace>.svc.cluster.local <instance name>-db
dbServerCertificate	Server	Management	
pgBouncerServerCertificate	Server	Management	
PGOTLSCertificate	Server	Management	
NATSTLSCertificate	Server	Management	
dbClientPostgres	Client	Management	
dbClientReplicator	Client	Management	
dbClientPgBouncer	Client	Management	
dbClientApicuser	Client	Management	

Certificate	Type (CA/Server/Client)	Subsystem	Notes
dbClientPrimaryuser	Client	Management	

Several certificates as noted above are required to be signed by the same CA as another certificate. For example, `portal-admin-client`, and `portal-admin`. This means that if the `portal-admin-client` certificate were to be customized, then the `portal-admin` certificate must also be customized, and signed by the same CA as `portal-admin-client`. To ensure that pairs of certificates like these are signed by the same CA, the Issuer for each certificate must be the same.

Generating custom certificates

You can generate custom certificates.

About this task

If you do not already have your own custom certificates that you can provide for the installation, you can manually prepare Certificate definitions for Cert-Manager to generate the required Certificates for you.

Note: If a certificate is signed by an internal or custom CA, include the full chain in the end certificate. If you omit the full chain, then a user who uses openssl to access the endpoint will see the following error: `error:num=20:unable to get local issuer certificate`

The following example shows the full chain for an end certificate (signed by a custom CA):

```
-----BEGIN CERTIFICATE -----
Cert contents for end-cert
-----END CERTIFICATE -----
-----BEGIN CERTIFICATE -----
Cert contents for Intermediate-CA
-----END CERTIFICATE -----
-----BEGIN CERTIFICATE -----
Cert contents for Root CA
-----END CERTIFICATE -----
```

Procedure

1. Create a Certificate Authority (CA) and an Issuer for Signing
 - a. Use the following YAML definition to generate a CA named `ingress-ca`, and an Issuer named `ingress-issuer` using that CA. Place the following definition into a file such as `ingress-ca.yaml`:

```
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
name: ingress-ca
spec:
secretName: ingress-ca
commonName: "ingress-ca"
usages:
- digital signature
- key encipherment
- cert sign
isCA: true
issuerRef:
name: selfsigning-issuer
kind: Issuer
duration: 87600h # 10 years
renewBefore: 720h # 30 days
privateKey:
rotationPolicy: Always
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
name: ingress-issuer
spec:
ca:
secretName: ingress-ca
```

- b. Apply this file into the cluster:

```
kubectl apply -f ingress-ca.yaml -n <namespace>
```

This will create a CA Certificate named `ingress-ca`, and also a Secret named `ingress-ca`. This CA will be used to sign the rest of the custom certificates.

- c. Verify the presence of both:

```
kubectl get certificate -n <namespace>
kubectl get secret -n <namespace>
```

2. Generating common subsystem communication certificates with Cert-Manager for management CR.
 - a. Use the following YAML definition to generate the following Common Certificates and Secrets: `analytics-ingestion-client`, and `portal-admin-client`. Place it into a file, for example, `common-certs.yaml`:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
name: analytics-ingestion-client
```

```

spec:
  commonName: analytics-ingestion-client
  secretName: analytics-ingestion-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
name: portal-admin-client
spec:
  commonName: portal-admin-client
  secretName: portal-admin-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
---

```

Note that the `issuerRef` uses the Issuer created in Step 1. These certificates must be signed by the CA associated with that Issuer, `ingress-ca`.

- b. Apply `common-certs.yaml` into the cluster:

```
kubectl apply -f common-certs.yaml -n <namespace>
```

This will create Client Certificates and Secrets named `analytics-ingestion-client`, and `portal-admin-client`.

- c. Verify the presence of all of these certificates and secrets:

```
kubectl get certificate -n <namespace>
kubectl get secret -n <namespace>
```

3. Generating Front-end/Endpoint Certificates with Cert-Manager

- a. Generate server certificates.

For each Endpoint in a subsystem CR, you can use the following YAML definition to generate a Server Certificate:

```

apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
name: <endpoint-host.secretName>
spec:
  commonName: <endpoint.host.secretName>
  secretName: <endpoint.host.secretName>
  dnsNames:
  - <endpoint.host.Name>
  issuerRef:
    name: ingress-issuer
  usages:
  - "server auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always

```

Replace `<endpoint.host.secretName>` with the `secretName` in the `host` block of the `Endpoint` in the CR. For example, if the `Endpoint` in the CR is:

```

cloudManagerEndpoint:
  hosts:
  - name: admin.example.com
    secretName: cm-endpoint

```

The Certificate YAML should be updated as follows:

```

---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
name: cm-endpoint
spec:
  commonName: cm-endpoint
  secretName: cm-endpoint
  dnsNames:
  - admin.example.com
  issuerRef:
    name: ingress-issuer
  usages:
  - "server auth"
  - "signing"
  - "key encipherment"

```



```
duration: 17520h # 2 years
renewBefore: 720h # 30 days
privateKey:
  rotationPolicy: Always
```

Note that the `issuerRef` uses the Issuer created in Step 1. This certificate will be signed by the CA associated with that Issuer, `ingress-ca`.

Place the edited yml into a file, e.g. `<subsystem>-server.yaml`.

b. Apply this file into the cluster:

```
kubectl apply -f <subsystem>-server.yaml.
```

This will create a Server Certificate named `<subsystem>-server`, and also a Secret named `<subsystem>-server`.

c. Check for the presence of both with:

```
kubectl get certificate -n <namespace>
kubectl get secret -n <namespace>
```

d. Repeat Step 3.a, step 3.b and step 3.c for each subsystem.

In each case, the Certificates must use the same Issuer created in Step 1. This ensures that all custom certificates are signed by the same CA Certificate.

Note that the `issuerRef` uses the Issuer created in Step 1. These certificates must be signed by the CA associated with that Issuer, `ingress-ca`.

e. Place the YAML definitions for each Certificate into a YAML file, for example, `endpoint-certs.yaml`. Apply this file into the cluster:

```
kubectl apply -f endpoint-certs.yaml -n <namespace>
```

This will create Client Certificates and Secrets for each Endpoint named `<endpoint.host.secretName>`.

f. Check for the presence of all of these with:

```
kubectl get certificate -n <namespace>
kubectl get secret -n <namespace>
```

4. Choose one of the following actions, based on whether or not you are customizing internal API certificates:

- If you do not plan to customize internal certificates, continue with [Configuring custom certificates before installation](#).
- If you want to generate some, or all, of the internal certificates, continue with the next step.

Note: It is not recommended to customize Internal certificates unless you have a specific requirement to do so.

5. Generate custom internal certificates using `custom-certs-internal.yaml`.

Each of the Internal certificates listed in [Custom certificates reference](#) have differing strict requirements for Common Names and DNS Names (i.e. Subject Alternative Names).

To generate custom certificates that meet these requirements, use the provided `custom-certs-internal.yaml` file, which can be found in the `helper_files` installation archive. For more information, see [Obtaining product files](#).

This file contains Certificate YAML definitions for all Internal certificates, CA, and Issuer that can be customized for each subsystem of IBM API Connect. The Certificate definitions file is already populated with the recommended default values.

a. When using custom internal certificates, you are required to specify a site name. The site name is used as the identifier for the PostgreSQL database Cluster used by API Connect. The site name chosen is added to the Management Subsystem CR YAML file before installing the Management Subsystem. In addition, this name is also required for use with the custom internal certificates, so a site name should be chosen now before proceeding.

b. Open `custom-certs-internal.yaml` in a text-editor, and make the following changes:

i. Search for the `pg-bouncer-server-certificate` certificate definition. Replace `commonName` and `dns-names` with the following values:

```
• commonName: management-pgo_cluster_id-postgres-pgbouncer
• dnsNames:
  - "*.<namespace>"
  - ".*.<namespace>.svc"
  - "management-pgo_cluster_id-postgres-pgbouncer.<namespace>.svc"
  - "management-pgo_cluster_id-postgres-pgbouncer"
```

ii. Find and replace each occurrence of `<namespace>` with the desired namespace for the deployment. This must be done even if the `default` namespace is to be used.

iii. Also in `custom-certs-internal.yaml`, replace each occurrence of `pgo_cluster_id` with the chosen site name described in Step 5.a.

c. Apply the file to the Cluster:

```
kubectl apply -f custom-certs-internal.yaml -n <namespace>
```

6. When all desired custom certificates are created and applied into the Kubernetes Cluster as `Certificates` and `Secrets`, update the Subsystem CR YAML files to use the custom certificates. See [Configuring custom certificates before installation](#).

Generating custom certificates using cert-manager in a two data center deployment

You can generate custom certificates in a two data center disaster recovery deployment on Kubernetes.

About this task

You can provide your own custom certificates when you deploy IBM API Connect, if required, instead of having the application generate and manage them for you. In most deployment cases to use the built in default Cert-Manager integration is the preferred method.

When you use the custom option, the use of Cert-Manager is the recommended method for generating custom certificates if that fits your business needs. However, use of Cert-Manager is not required, and custom certificates can be generated in any way that suit your needs.

The following example of providing custom certificates uses Cert-Manager as a model for generating them. For generating custom certificates, the minimum supported version of cert-manager is v1.5.1.

Procedure

1. Set `KUBECONFIG` for the target cluster:

```
export KUBECONFIG=<path_to_cluster_config_YAML_file>
```

Example path:

```
/Users/user/.kube/clusters/<cluster_name>/kube-config-<cluster_name>.yaml
```

2. Install cert-manager version 1.5.1 or later :
 - a. Download v1.9.1 from <https://github.com/cert-manager/cert-manager/releases/tag/v1.9.1>.
 - b. Install cert-manager. Do not specify a namespace:

```
kubectl apply -f cert-manager.yaml
```

- c. Check the status of the `cert-manager` pods:

```
kubectl -n cert-manager get po
```

Wait for `cert-manager` pods to enter `Running 1/1` status before proceeding. There are 3 `cert-manager` pods in total.

3. Set up the customized external certificates:

For quick setup, you can use the provided yaml files that are included in the `helper_files` that you expanded in [Deploying operators and cert-manager](#). The yaml for the active data center (DC1) is called `custom-certs-external-dc1.yaml` and the yaml for the warm-standby data center (DC2) is called `custom-certs-external-dc2.yaml`.

- a. Open `custom-certs-external-dc1.yaml` and `custom-certs-external-dc2.yaml` in a text-editor. Find and replace each occurrence of `example.com` with the desired ingress subdomain for the API Connect stack.
- b. Ensure that the management CR for DC1 is ready for applying but not yet applied, make a note of the `hostname` of the site.
- c. In `custom-certs-external-dc1.yaml` find `spec.dnsNames`

For example, in DC1:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: dc1-mgmt-replication
spec:
  commonName: dc1-mgmt-replication
  secretName: dc1-mgmt-replication
  dnsNames:
  - <INSERT SITE HOSTNAME HERE. THIS WILL BE THE SAME VALUE AS IN THE MANAGEMENT CR spec.multiSiteHA.hosts.name for site 1>
  issuerRef:
    name: ingress-issuer
  usages:
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
```

- d. Replace the line that begins - `<INSERT SITE HOSTNAME HERE` with - `hostname of the site`.
- e. Ensure that the management CR for DC2 is ready for applying but not yet applied, make a note of the `hostname` of the site.
- f. In `custom-certs-external-dc2.yaml` find `spec.dnsNames` and replace the line that begins - `<INSERT SITE HOSTNAME HERE` with - `hostname of the site`.
- g. Ensure that the portal CR for DC1 is ready for applying but not yet applied, make a note of the `hostname` of the site.
- h. In `custom-certs-external-dc1.yaml` find `spec.dnsNames`

For example, in DC1:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: dc1-ptl-replication
spec:
  commonName: dc1-ptl-replication
  secretName: dc1-ptl-replication
  dnsNames:
  - <INSERT SITE HOSTNAME HERE. THIS WILL BE THE SAME VALUE AS IN THE PORTAL CR spec.multiSiteHA.hosts.name for site 1>
  issuerRef:
    name: ingress-issuer
  usages:
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
```

- i. Replace the line that begins - `<INSERT SITE HOSTNAME HERE` with - `hostname of the site`.
- j. Ensure that the portal CR for DC2 is ready for applying but not yet applied, make a note of the `hostname` of the site.

- k. In `custom-certs-external-dc2.yaml` find `spec.dnsNames` and replace the line that begins - `<INSERT SITE HOSTNAME HERE` with - `hostname of the site`.
4. Use the following steps to allow `ingress-ca` secrets to be the same on both data centers.
- On DC1 apply the file `custom-certs-external-dc1.yaml`:


```
kubectl -n <namespace> apply -f custom-certs-external-dc1.yaml
```
 - Validate that the command succeeded:


```
kubectl get certificates -n <namespace>
```
 - Export `ingress-ca` secret as a yaml from DC1:


```
kubectl -n <namespace> get secret ingress-ca -o yaml > ingress-ca.yaml
```
 - Edit the `ingress-ca.yaml` file to remove all labels, annotations, `creationTimestamp`, `resourceVersion`, `uid`, and `selfLink`.
 - Copy the `ingress-ca.yaml` from DC1 to DC2 and apply that file on DC2:


```
kubectl -n <namespace> apply -f ingress-ca.yaml
```
 - On DC2 apply the file `custom-certs-external-dc2.yaml`:


```
kubectl -n <namespace> apply -f custom-certs-external-dc2.yaml
```
 - Use the following commands to test that they are the same, on DC1 run:


```
kubectl -n <namespace> get secrets ingress-ca -o yaml | grep tls.crt | awk '{print $2}' | base64 -d > /tmp/ingress.pem.dc1
```
 - On DC2 run:


```
kubectl -n <namespace> get secrets ingress-ca -o yaml | grep tls.crt | awk '{print $2}' | base64 -d > /tmp/ingress.pem.dc2
```
 - To see the differences run:


```
diff /tmp/ingress.pem.dc1 /tmp/ingress.pem.dc2
```

The files should be the same.
 - To ensure that the certificates are working correctly and that they are using the `ingress-ca` secret. First, get the `portal-admin-client crt` file:


```
kubectl -n <namespace> get secrets portal-admin-client -o yaml | grep tls.crt | awk '{print $2}' | base64 -d > /tmp/admin-client.crt
```
 - Test that it is working by using OpenSSL:


```
openssl verify -verbose -CAfile /tmp/ingress.pem.dc1 /tmp/admin-client.crt
```

If it is working, you should see:

```
/tmp/admin-client.crt: OK
```
5. Update the API Connect subsystem CR .yaml files to use the custom certificates. To do this, continue with [Configuring custom certificates before installation](#).

Configuring custom certificates before installation

Configure custom certificates for your API Connect deployment before you begin the installation procedures.

About this task

To deploy API Connect with custom certificates, complete the following steps to configure your custom certificates before applying any subsystem installation CRs to the cluster.

Procedure

- Determine which certificates you want to customize.

For a list of certificates that can be customized, see [Custom certificates reference](#).

You can customize the following types of certificates:

 - Endpoint (External Frontend/Ingress) certificates:

These are the certificates for external, public-facing endpoints where users interact with API Connect. You can customize none, some, or all of the endpoint certificates.

If you initially deploy APIC using default certificates for endpoints, you can convert them to custom certificates later as explained in [Converting to custom front-end/ingress certificates after deployment](#).
 - Internal (CA, Client, Server, and Common Subsystem Communication) certificates:

These are the certificates used within API Connect, so that subsystems can communicate. If you want to customize any of the internal certificates, then you must customize all of them; however you don't have to create them all yourself. Any custom certs that are not specifically named in your deployment CR will be generated automatically during installation.

If you initially deploy API Connect using default certificates for internal certificates, you cannot convert them to custom certificates later.
- Prepare your custom certificates.

If you don't have a set of custom certificates already, you can use Kubernetes Cert-manager to generate them as explained in [Generating custom certificates](#).

When preparing your custom certificates, remember that certain pairs of certificates must be signed by the same Certificate Authority, as noted in [Custom certificates reference](#).

If you're installing a two data center disaster recovery configuration, and are using custom certificates, then the subject names of any client certificates must be the same in both data centers, as noted in [Custom certificates reference](#). For example, both data centers subject name could be `CN=portal-admin-client`, or they could both be `CN=pt1-adm-client, O=cert-manager`, but they must be identical.

3. If needed, create a Kubernetes secret for each custom certificate.

Each custom certificate that you use must be captured in a TLS secret. To create a secret, use the following command:

```
kubectl create secret generic my-tls-secret --from-file=tls.crt=<path_to_cert_file> --from-file=tls.key=<path_to_key_file>
--from-file=ca.crt=<path_to_cacert_file>
```

When you create each secret, it's helpful to reference its purpose or its corresponding certificate in the secret name, to ensure you specify the correct secret for each custom certificate.

4. Edit the `ingress-issuer-v1.yaml` file and delete the default definitions for the certificates that you will customize.

The `ingress-issuer-v1.yaml` file is provided as part of the installation `helper_files` archive.

This file is used for generating the default certificates. To use a custom certificate, you must delete the default definition to ensure that API Connect uses your certificate instead of the default certificate. If you customize all of the default certificates and delete those definitions from the file, then you don't need to use the file during installation.

5. Customize endpoint (External Frontend/Ingress) certificates in the corresponding subsystem CRs.

For each custom endpoint certificate, delete the annotation that specifies the default issuer, and replace the secret name with your own custom secret name.

- a. Open the subsystem CR for editing and locate the endpoint definition.

For example, the following snippet shows the definition for the `cloudManagerEndpoint` in the Management CR:

```
cloudManagerEndpoint:
  annotations:
    cert-manager.io/issuer: ingress-issuer
  hosts:
  - name: admin.example.com
    secretName: cm-endpoint
```

- b. Delete the annotation for the `cert-manager.io/issuer`.

Delete the field name as well as the value; you can leave the `annotations:` label in place.

- c. Replace the default secret name with the name of the Kubernetes secret that you created for that endpoint's custom certificate.

In the following example, `cm-endpoint` is the default secret name, which must be replaced. Make sure to reference the correct secret for the custom certificate.

```
cloudManagerEndpoint:
  annotations:
  hosts:
  - name: admin.example.com
    secretName: my_cm-endpoint
```

6. Customize internal (CA, Client, Server, and Common Subsystem Communication) certificates.

- a. Open the subsystem CR for editing.

- b. Set the `microServiceSecurity` setting to `custom`.

By default, the setting is configured for `certManager` as shown in the following example:

```
microServiceSecurity: certManager
certManagerIssuer:
  name: selfsigning-issuer
  kind: Issuer
```

- c. Delete (or comment-out) the `certManagerIssuer` block.

In the following example, `microServiceSecurity` is set to `custom` and the `certManagerIssuer` block is commented out:

```
microServiceSecurity: custom
#certManagerIssuer:
#  name: selfsigning-issuer
#  kind: Issuer
```

7. Add the `customCertificates` block and list the secret name for each internal custom certificate that you created for the subsystem.

Each subsystem supports a different set of Common Subsystem Communication and External Frontend/Ingress certificates. You do not have to provide a custom version of every certificate supported by the subsystem. Just list the custom certificates that you created; any certificates that you omit from the list will be generated automatically during installation.

For example, your `customCertificates` list for the Management subsystem might look like the following example:

```
customCertificates:
- name: caCertificate
  secretName: my_ingress-ca
- name: clientCertificate
  secretName: my_management-client
- name: serverCertificate
  secretName: my_management-server
- name: dbServerCertificate
  secretName: my_db-server-certificate
- name: pgBouncerServerCertificate
  secretName: my_pg-bouncer-server-certificate
- name: PGOTLSCertificate
  secretName: my_pgo.tls
- name: NATSTLSCertificate
  secretName: my_management-natscluster-mgmt
```

```

- name: dbClientPostgres
  secretName: my_db-client-postgres
- name: dbClientReplicator
  secretName: my_db-client-replicator
- name: dbClientPgouncer
  secretName: my_db-client-pgouncer
- name: dbClientApicuser
  secretName: my_db-client-apicuser
- name: dbClientPrimaryuser
  secretName: my_db-client-primaryuser

```

What to do next

Deploy API Connect:

- If you did not deploy the API Connect operator yet, proceed to [Deploying operators in a single-namespace API Connect cluster](#) or [Deploying operators in a multi-namespace API Connect cluster](#).
- If you already deployed the operator, proceed to [Installing the API Connect subsystems](#).

Converting to custom front-end/ingress certificates after deployment

Convert a front-end certificate generated by cert-manager to a custom certificate for an existing IBM® API Connect deployment.

About this task

If you deploy IBM API Connect using certificates that were generated by cert-manager, you can later convert front-end (ingress) certificates to use custom certificates instead. For a list of the certificates that are generated and managed by cert-manager, see [Custom certificates reference](#).

Restriction: This task applies only to front-end certificates and is not supported for the "Common Subsystem Communication" and "Internal" type certificates.

Procedure

1. Create the secrets you want to use with the `kubectl create secret` command.
2. Edit the subsystem CR, and make the following changes for the endpoint secret you want to change:
 - In the `annotations` section, remove the following line:

```
cert-manager.io/issuer: ingress-issuer
```

- In the `hosts` section, update the `secretName` to the new secret that you created in step 1.

The following example shows where the updates should be made:

```

spec:
  <endpoint>
    annotations:
      cert-manager.io/issuer: ingress-issuer REMOVE THIS LINE
    hosts:
      - name: <api endpoint hostname>
        secretName: CHANGE THIS TO THE NEW SECRET

```

Webhooks deployed and managed by the API Connect Operator

The API Connect Operator deploys and manages various Kubernetes API webhooks in the cluster to assist in management of its custom resources.

The webhooks included admission controllers and conversion webhooks.

Defaulting webhook

- Purpose
The defaulting webhook is a type of [Mutating Admission Webhook](#) which runs against Custom Resources (CRs) at **CREATE** or **UPDATE** time to populate default values in the CR's `spec`.
- Creation
Currently the defaulting webhook runtime is part of the API Connect Operator runtime, that is it exists within the API Connect Operator pod and container. When the operator boots, it creates `MutatingWebhookConfiguration` resources for each CRD (listed below). This `MutatingWebhookConfiguration` is a cluster-scope resource, but the configuration within will be specific to the namespace and operator instance which created it. If more than one instance of the API Connect Operator is deployed across a cluster (in different namespaces) there will be a set of webhook configurations for each API Connect Operator instance and namespace.

The name for the `MutatingWebhookConfiguration` resource for each CRD will take the following form:

```
<namespace>.<crd-name>.defaulter[.<component>].apiconnect.ibm.com
```

The following CustomResourceDefinitions utilize a defaulting webhook:

```

AnalyticsCluster
ApiconnectCluster
GatewayCluster
ManagementBackup

```

```
ManagementCluster
ManagementDBUpgrade
ManagementRestore
PortalCluster
```

- Lifecycle

While the defaulting webhook runtime itself lives within the API Connect Operator pod, the **MutatingWebhookConfiguration** is a cluster-scope resource and is stand-alone. When it is created, an **ownerReference** is set on it that references the API Connect Operator's **ClusterRole**. Thus, the lifecycle of the **MutatingWebhookConfiguration** resources will be linked with the **ClusterRole**, which is linked with the installation of the operator itself.

When the operator is uninstalled, the **MutatingWebhookConfiguration** resources for that operator instance are also removed.

- Example

The below output shows the defaulting webhook configurations created for an operator deployed in the **apiconnect-operator** namespace. One can be seen for each CRD.

```
$ oc get mutatingwebhookconfiguration
NAME
apiconnect-operator.analyticsclusters.default.analytics.apiconnect.ibm.com 1 4d4h
apiconnect-operator.apiconnectclusters.default.apiconnect.ibm.com 1 4d4h
apiconnect-operator.gatewayclusters.default.gateway.apiconnect.ibm.com 1 4d4h
apiconnect-operator.managementbackups.default.management.apiconnect.ibm.com 1 4d4h
apiconnect-operator.managementclusters.default.management.apiconnect.ibm.com 1 4d4h
apiconnect-operator.managementdbupgrades.default.management.apiconnect.ibm.com 1 4d4h
apiconnect-operator.managementrestores.default.management.apiconnect.ibm.com 1 4d4h
apiconnect-operator.portalclusters.default.portal.apiconnect.ibm.com 1 4d4h
```

Validating webhook

- Purpose

The validating webhook is a type of **ValidatingAdmissionWebhook** which runs against Custom Resources (CRs) at **CREATE** or **UPDATE** time to validate the CR contains a configuration which will yield an operable cluster. The CustomResourceDefinitions use OpenAPI schema validation for basic property checks (such as integer bounds and non-empty strings), but more complex validation of the configuration is done via the webhook.

- Creation

Currently the validating webhook runtime is part of the API Connect Operator runtime, i.e. it exists within the API Connect Operator pod and container. When the operator boots, it creates **ValidatingWebhookConfiguration** resources for each CRD (listed below). This **ValidatingWebhookConfiguration** is a cluster-scope resource, but the configuration within will be specific to the namespace and operator instance which created it. If more than one instance of the API Connect Operator is deployed across a cluster (in different namespaces) there will be a set of webhook configurations for each API Connect Operator instance and namespace.

The name for the **ValidatingWebhookConfiguration** resource for each CRD will take the following form:

```
<namespace>.<crd-name>.validator[.<component>].apiconnect.ibm.com
```

The following CustomResourceDefinitions utilize a validating webhook:

```
AnalyticsCluster
ApiconnectCluster
GatewayCluster
ManagementBackup
ManagementCluster
ManagementDBUpgrade
ManagementRestore
PortalCluster
```

- Lifecycle

While the validating webhook runtime itself lives within the API Connect Operator pod, the **ValidatingWebhookConfiguration** is a cluster-scope resource and is stand-alone. When it is created, an **ownerReference** is set on it that references the API Connect Operator's **ClusterRole**. Thus, the lifecycle of the **ValidatingWebhookConfiguration** resources will be linked with the **ClusterRole**, which is linked with the installation of the operator itself.

When the operator is uninstalled, the **ValidatingWebhookConfiguration** resources for that operator instance are also removed.

- Example

The below output shows the validating webhook configurations created for an operator deployed in the **apiconnect-operator** namespace. One can be seen for each CRD.

```
validatingwebhookconfiguration
NAME
apiconnect-operator.analyticsclusters.validator.analytics.apiconnect.ibm.com 1 4d4h
apiconnect-operator.apiconnectclusters.validator.apiconnect.ibm.com 1 4d4h
apiconnect-operator.gatewayclusters.validator.gateway.apiconnect.ibm.com 1 4d4h
apiconnect-operator.managementbackups.validator.management.apiconnect.ibm.com 1 4d4h
apiconnect-operator.managementclusters.validator.management.apiconnect.ibm.com 1 4d4h
apiconnect-operator.managementdbupgrades.validator.management.apiconnect.ibm.com 1 4d4h
apiconnect-operator.managementrestores.validator.management.apiconnect.ibm.com 1 4d4h
apiconnect-operator.portalclusters.validator.portal.apiconnect.ibm.com 1 4d4h
```

Installing API Connect

Use these instructions to install a deployment of API Connect on Kubernetes, OpenShift, and IBM Cloud Pak for Integration.

To deploy API Connect, complete the tasks in the order below.

- **Deployment requirements**

Describes the system requirements for deploying into the Kubernetes, OpenShift, and Cloud Pak for Integration runtime environments.

- [Deployment procedures](#)

You can use these procedures to deploy API Connect on either native Kubernetes or OpenShift.

Deployment requirements

Describes the system requirements for deploying into the Kubernetes, OpenShift, and Cloud Pak for Integration runtime environments.

Before you begin

Note: This article refers to third-party software that IBM does not control. As such, the software might change and this information might become outdated. Kubernetes is a platform for automated deployment, scaling, and operation of application containers across clusters of hosts, providing container-centric infrastructure. For more information, see <https://kubernetes.io>.

These instructions assume you have a working Kubernetes environment and understand how to manage Kubernetes.

About this task

Review all the requirements listed in the following steps before starting your API Connect deployment.

Note: If you plan to install the Automated API behavior testing application, review the *Before you begin* section in [Installing the Automated API behavior testing application](#) because the requirements might have an impact on your deployment choices for API Connect.

Procedure

1. Verify that you have supported software.
 - Refer to the IBM Software Product Compatibility Reports site for detailed requirements for operating systems, supporting software, and other prerequisites. See [Detailed system requirements for a specific product](#)
 - For new installations, the API Connect operators and operands must be from the same version of API Connect. This requirement applies to native Kubernetes and OpenShift installations.
 - Note that the API Connect operator version *number* does not match the API Connect release version *number*. You can view a mapping of API Connect release number to API Connect operator version numbers in [Deploying operators and cert-manager](#).
 - For API Connect operands (Management, Portal, Analytics, and Gateway subsystems), the version number matches the API Connect release version number.

2. Verify that your Kubernetes environment is running and meets the configuration requirements for API Connect:
 - When using IBM Cloud Kubernetes Service, ingress-nginx is required.
 - Specify settings for the ingress controller for a Kubernetes runtime environment. See [Kubernetes ingress controller prerequisites](#).
 - The timezone for API Connect pods is set to UTC. Do not change the timezone for the pods.
 - To ensure the API Connect services have time to start, we recommend increasing the *proxy-read-timeout* and *proxy-send-timeout* values in the *config.map* used to configure the `kubernetes/ingress-nginx` ingress controller. The following settings should be increased:
 - `proxy-read-timeout: "240"`
 - `proxy-send-timeout: "240"`240 seconds (4 minutes) is a recommended minimum; actual value will vary depending upon your environment. If there is a load balancer in front of the worker node(s), then the load balancer configuration may also need to have extended timeouts.
 - Storage type:
 - When selecting storage type, review the following considerations:
 - Management subsystem requires block storage.
 - Developer Portal subsystem supports block storage or local volume storage.
 - Analytics subsystem supports block storage or local volume storage.

Supported storage types

- Block storage
 - Select a block storage format of your choosing in order for API Connect components to be supported. Supported block storage types include, but are not limited to:
 - Rook Ceph
 - Note: `cephrbd` (RADOS Block Device) is recommended over `cephfs` file system
 - OpenShift Container Storage
 - Portworx
 - IBM Cloud Block Storage
 - AWS: the `gp2`, `gp3`, `io1`, and `io2` types of Elastic Block Storage (EBS) are supported.
 - IBM Spectrum Scale with Block Storage
 - Azure Premium Storage
 - GCE Persistent Disk

See also the [API Connect v10 WhitePaper 1.5](#) for information on limitations observed with other block storage solutions:

<https://community.ibm.com/community/user/integration/viewdocument/api-connect-deployment-whitepaper-v?CommunityKey=210cca0-a9f9-45c6-9b28-01a28f4ce947&tab=librarydocuments&LibraryFolderKey=&DefaultView=folder>

- Local volume storage

Management subsystem and Developer Portal subsystem

All subsystems will run on local volume storage but it is not recommended for the Management subsystem and Developer Portal subsystem because it has severe performance impacts for high availability deployments, and for pod allocation.

Note also that local volume storage is not available in most cloud environments, and might incur limitations in disaster recovery scenarios, since it is tied to the node on which the pod is run. This means that use of LVS might be adequate for deployments with a one

replicaprofile, but inadequate for deployments with a three replica profile.

Analytics subsystem

For Analytics subsystems, transactions per second (TPS) is three times faster with local volume storage than with Ceph block storage. In a production scenario, you will have multiple analytics-storage (OpenSearch) nodes forming a storage cluster. OpenSearch will manage data replication across its cluster and ensure high availability, thus mitigating possible local volume storage drawbacks with disaster recovery.

Non-supported storage

- NFS
API Connect v10 cannot be deployed on NFS.
 - GlusterFS
GlusterFS is not recommended as a storage option due to severe performance degradation and in some cases, loss of data. There are known GlusterFS issues that affect OpenSearch (which is used in the API Connect analytics service), the Developer Portal and the Management subsystem
- Storage class configuration:
In your Kubernetes cluster configuration, you must create storage classes with the setting:

```
volumeBindingMode: WaitForFirstConsumer
```

This setting is needed to avoid volume node **affinity** conflicts, such as those caused by **requiredPodAntiAffinity** restrictions. If **WaitForFirstConsumer** is the binding mode, PVCs (PersistentVolumeClaims) are only bound once the pod is actually created.

By contrast, when the binding mode is immediate (**volumeBindingMode: Immediate**), the PVCs are bound to PVs (PersistentVolumes) as soon as the PV is created.

- Example:
The Postgres operator creates the PVC first and then deploys the deployments. In immediate mode, PVCs are bound to PVs as soon as the PV is created. The **pgbackrest** pod starts successfully, but because **requiredPodAntiAffinity** restrictions are set, and the **postgres** pod is **NOT ALLOWED** to schedule on the same node as **pgbackrest** pod, Kubernetes cannot schedule the pod in node **eu-west-1c**. Also, because the volume attached to the pod is at **eu-west-1c**, Kubernetes cannot move the pod to any other node, thus causing **volume node affinity** conflict.
- Example:
When deploying to a multi-zone cluster, you must use StorageClass with **WaitForFirstConsumer** because when binding mode is immediate, successful deployment is indeterminate because of interactions between pod/node affinity and the Kubernetes provisioning of the PV on the right node where the pod is being provisioned.

For more information, see "Volume Binding Mode" on <https://kubernetes.io/docs/concepts/storage/storage-classes/>.

Note: During deployment, you are required to specify the name of the storage class you created. You can either use the setting **storageClassName: \$STORAGE_CLASS** in the applicable CR (such as apiconnect top-level CR or an individual subsystem CR), or you can use a configuration UI such as Platform Navigator UI or OpenShift Container Platform (OCP) UI.

- When the Analytics service is configured to store data, (that is, it is not configured for ingestion-only), the service depends on OpenSearch, which requires map counts higher than the operating system defaults. See: [Installing the Analytics subsystem](#).
If you are deploying on OpenShift, the map count is preconfigured (**vm.max_map_count = 1027580**) with a value that is sufficiently large for OpenSearch so you do not need to modify this setting.
- Ensure that your Kubernetes deployment addresses known security vulnerabilities with SSH to prevent the use of weak SSH Message Authentication Code (MAC) and Key exchange algorithms (KexAlgorithms) with TCP over port 22. Use of weak algorithms can allow an attacker to recover the plain text message from the encrypted text. If you have not yet addressed the security vulnerabilities with weak MACs and KexAlgorithms, update your configuration manually by adding the following lines to `/etc/ssh/sshd_config`:

```
KexAlgorithms curve25519-sha256@libssh.org,diffie-hellman-group-exchange-sha256
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,umac-128@openssh.com
```

- IBM API Connect requires a number of container registry repositories to be added to your container registry service if automatic repository creation is not supported. Follow the procedure in Step 4 of [Obtaining product files](#).
- Logging:
 - Configure a remote server for logging. See [Logging](#). Pods are terminated during an upgrade, and the logs will be lost if not stored remotely.
 - Configure the log rotation by setting the **max-size** and **max-files** values to a setting appropriate for your deployment. In a Kubernetes deployment, insufficient values for max-size and max-files can cause logs to grow too large and eventually force pods to restart.
For more information, visit the Docker documentation and search for "JSON File logging driver".

3. Ensure your deployment meets the DNS requirements for API Connect domain names, host names, and endpoints.

You specify endpoints when you edit the custom resource templates for each subsystem, and when you use the Cloud Manager UI to configure the services for your cloud. The domain names you use for the endpoints must be recognized by your DNS configuration. DNS entries must meet these requirements:

Important: For API Connect on native Kubernetes and OpenShift, do not use coredns 1.8.1 through 1.8.3. These releases have a defect that can prevent pods for Developer Portal from starting. See <https://coredns.io/>.

- Endpoints cannot contain the underscore character "_" or uppercase characters.
- Host names that are used to create endpoints cannot contain the underscore character "_" or uppercase letters.
- DNS must be configured with domain names that correspond to the endpoints configured at installation time for each subsystem.
- Host names and DNS entries may not be changed on a cluster after the initial installation.
- Kubernetes ingress limits the character set for DNS names to not support the underscore character "_". This means you cannot specify underscores in domain names that are used as endpoints. For example, **my_domain.abc.com** and **my.domain_abc.com** are not supported for `<xxx>.<hostname>.<domainname>`, and will cause an error:

```
Invalid value: "my_domain.abc.com": a DNS-1123 subdomain must consist of lowercase alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character (e.g. 'example.com', regex used for validation is '[a-z0-9]([-a-z0-9]*[a-z0-9])?(\.[a-z0-9]([-a-z0-9]*[a-z0-9])?)*')
```


- We recommend that you keep a record of the DNS entries and endpoint names, as you will need to map the DNS entries and use the same endpoints when restoring a backup of the management database. Note that endpoints are FQDNs, entered in all lowercase. For an overview on endpoints, review [Deployment overview for endpoints and certificates](#)
- Specifying domain names when deploying CRs (custom resources) for each subsystem:
The custom resource templates provide endpoint settings that consist of a default endpoint prefix and a variable `$$STACK_HOST` that you set to specify your domain name, as part of [Installing the API Connect subsystems](#).

Table 1. Endpoints with domain names that must be configured in DNS

Default endpoint name	Description
<code>admin.\$\$STACK_HOST</code>	Cloud Manager URL endpoint for the Management subsystem
<code>manager.\$\$STACK_HOST</code>	API Manager URL endpoint for the Management subsystem
<code>api.\$\$STACK_HOST</code>	Platform REST API Endpoint for admin and provider APIs
<code>consumer.\$\$STACK_HOST</code>	Platform REST API Endpoint for consumer APIs for the Management subsystem.
<code>rgw.\$\$STACK_HOST</code>	Gateway Endpoint for API invocation for DataPower API Gateway
<code>rgwd.\$\$STACK_HOST</code>	Gateway Management Endpoint for DataPower API Gateway
<code>gw.\$\$STACK_HOST</code>	Gateway Endpoint for API invocation for DataPower Multi-Protocol Gateway (v5-compatible)
<code>gwd.\$\$STACK_HOST</code>	Gateway Management Endpoint for DataPower Multi-Protocol Gateway (v5-compatible)
<code>ai.\$\$STACK_HOST</code>	Analytics ingestion endpoint for the Analytics subsystem
<code>api.portal.\$\$STACK_HOST</code>	Developer Portal admin endpoint
<code>portal.\$\$STACK_HOST</code>	Developer Portal UI endpoint

- Specifying endpoints using the Cloud Manager interface
After initial deployment of the Kubernetes custom resource for each subsystem, you will also enter the endpoints in the Cloud Manager user interface when defining each service as part of [Defining your topology](#).
- The Cloud Provider for the target Kubernetes platform must provide Ingress Controller support for SSL Passthrough.
For example, if the Cloud Provider uses or suggests the use of the NGINX Community Ingress, then the `--enable-ssl-passthrough` flag would need to be provided at runtime, as described at <https://kubernetes.github.io/ingress-nginx/user-guide/tls/#ssl-passthrough>. For IBM Cloud Kubernetes Service, see https://cloud.ibm.com/docs/containers?topic=containers-ingress-user_managed.
 - Determine your strategy for using certificates and cert-manager with API Connect.
If you haven't already, review the topics in the section [Certificates in a Kubernetes environment](#).

Consider the following:

- Will you use a cert-manager (recommended) or manually manage your certificates?
API Connect provides a cert-manager and default certificates. If you use these, you can follow default configuration steps. See [Deploying operators and cert-manager](#).
 - Will you use custom certificates rather than default certificates? See [Custom certificates on Kubernetes](#).
- If you plan to deploy API Connect on Kubernetes with DataPower Gateway in a *non-Kubernetes* environment, such as a DataPower appliance, review the API Connect requirements for operation with DataPower Gateway:
 - When deploying with a DataPower appliance, configure a Gateway Service Management Endpoint and API invocation Endpoint in DataPower. See [Configuring the API Connect Gateway Service](#) in the appropriate version of the [DataPower documentation](#).
 - Ensure that DataPower Gateway firmware version you plan to install is compatible with the API Connect Management server version.
Note:
You can use any combination of API Connect 10.0.5.x with DataPower Gateway 10.5.0.x or DataPower API Gateway 10.5.0.x.

Before you install, best practice is to review the latest compatibility support for your version of API Connect. To view compatibility support, follow the instructions in [IBM API Connect Version 10 software product compatibility requirements](#) to access API Connect information on the Software Product Compatibility Reports website. Once you access information for your version of API Connect, select Supported Software, Integration Middleware, and view the list of compatible DataPower Gateway versions.

- If you plan to deploy the API Connect management subsystem and the DataPower Gateway subsystem on a Kubernetes deployment, managed by the DataPower and API Connect Operators, you can use any combination of API Connect 10.0.5.x with DataPower Gateway 10.5.0.x or DataPower API Gateway 10.5.0.x.
- Observe namespace restrictions:
 - OpenShift
Only one top level CR (APIConnectCluster) can be deployed in each namespace.
 - Cloud Pak for Integration
In the Platform Navigator UI, only one APIConnect capability can be deployed in each namespace.
 - Native Kubernetes
Only one instance of a subsystem can be deployed in each namespace. Deploying multiple similar subsystems in a single namespace is not supported. For example, there is no support for deploying two management subsystems, `mgmt1`, `mgmt2` in a single namespace where apiconnect operator is installed.
- The following kernel or Kubernetes node limits for the Developer Portal subsystem must be set correctly:
 - Ensure that your kernel or Kubernetes node has the value of its `inotify` watches set high enough so that the Developer Portal can monitor and maintain the files for each Developer Portal site. If set too low, the Developer Portal containers might fail to start or go into a `non-ready` state when this limit is reached. If you have many Developer Portal sites, or if your sites contain a lot of content, for example, many custom modules and themes, then a larger number of `inotify` watches are required. You can start with a value of 65,000, but for large deployments, this value might need to go up as high as 1,000,000. The Developer Portal containers take `inotify` watches only when they need them. The full number is not reserved or held, so it is acceptable to set this value high.
 - Ensure that your kernel or Kubernetes node has a value of `nproc` (maximum number of processes) that applies to the user ID of the Portal pods that have been assigned, and that it is high enough to allow all of the Portal processes to execute. For smaller installations this might be as low as 16384, but for larger installations that have more concurrent web calls, you might need as many as 125205. If the number is too low, you will see errors like `"fork: retry: Resource temporarily unavailable"` in the Portal logs. Note that this value might need to be even higher if other, non-Portal, pods are sharing the same user ID.
 - Ensure that your kernel or Kubernetes node has a value of `nofiles` (maximum number of open file descriptors) that applies to the user ID of the Portal pods that have been assigned, and that it is high enough to allow the Portal to open all of the files that it requires. For smaller installations this might be as low as 16384, but for larger installations that have more concurrent web calls and Portal web sites, you might need as many as 1048576. If the number is too low,

you will see errors like "too many open files" in the Portal logs. Note that this value might need to be even higher if other, non-Portal, pods are sharing the same user ID.

- [IBM API Connect Version 10 software product compatibility requirements](#)
Ensure that you install the minimum API Connect operating system requirements. Use the IBM Software Product Compatibility Reports site to generate a requirements report appropriate for your API Connect version and environment.
- [API Connect licenses](#)
When you install a new version of API Connect, you must accept a license for the API Connect program that you purchased.

IBM API Connect Version 10 software product compatibility requirements

Ensure that you install the minimum API Connect operating system requirements. Use the IBM® Software Product Compatibility Reports site to generate a requirements report appropriate for your API Connect version and environment.

- [Generating a Software Product Compatibility Report](#)
- [Supported versions of Kubernetes](#)
- [Upgrading to an API Connect release that uses a newer version of Kubernetes](#)
- [Supported versions of OpenShift](#)
- [Upgrading to an API Connect release that uses a newer version of OpenShift](#)

Generating a Software Product Compatibility Report

To generate an API Connect requirements report, complete the following steps:

1. Open the [Detailed system requirements for a specific product](#) page on the IBM Software Product Compatibility Reports site.
2. Search for the IBM API Connect product.
3. In the Search results list, select IBM API Connect.
4. From the Version list, select the required version.
5. Use the Filters to refine the contents of the requirements report.
6. Click Submit to generate your requirements report.

Supported versions of Kubernetes

The following tables provide a quick reference of the supported Kubernetes versions for each API Connect release:

Table 1. API Connect and Kubernetes (K8S) compatibility matrix

API Connect version	K8S 1.20	K8S 1.21	K8S 1.22	K8S 1.23	K8S 1.24	K8S 1.25	K8S 1.26	K8S 1.27
10.0.1.7		✓	✓	✓				
10.0.1.8			✓	✓				
10.0.1.9				✓	✓	✓		
10.0.1.11				✓	✓	✓		
10.0.1.12					✓	✓	✓	
10.0.1.15						✓	✓	✓
10.0.4.0-ifix2 or later	✓	✓	✓					
10.0.5.0			✓	✓				
10.0.5.1			✓	✓				
10.0.5.2				✓	✓	✓		
10.0.5.3					✓	✓	✓	
10.0.5.4					✓	✓	✓	✓

Upgrading to an API Connect release that uses a newer version of Kubernetes

When you upgrade API Connect, both your current deployment and your target release must support the same version of Kubernetes. After the API Connect upgrade, you can optionally update Kubernetes to the highest version supported by the new release of API Connect.

It is possible you will need to upgrade both Kubernetes and API Connect more than once to complete the process. For example, suppose that your current deployment is running API Connect 10.0.1.7 on Kubernetes v1.21 and you want to upgrade to API Connect 10.0.5.4. The highest version of Kubernetes supported with API Connect 10.0.1.7 is v1.23, but the lowest version of Kubernetes supported by API Connect 10.0.5.4 is v1.24, so a direct upgrade of API Connect is not possible.

In this scenario, you must make multiple upgrades of both Kubernetes and API Connect. For example, one option is to use API Connect 10.0.1.11 on Kubernetes v1.23 as your interim release by completing the following upgrades:

1. On your API Connect 10.0.1.7 deployment, upgrade Kubernetes from v1.21 to v1.22, and then to v1.23.
2. Upgrade API Connect to 10.0.1.11 on Kubernetes v1.23.
3. On the API Connect 10.0.1.11 deployment, upgrade Kubernetes to v1.24.
4. Upgrade API Connect to 10.0.5.4 on Kubernetes v1.24.
5. On the API Connect 10.0.5.4 deployment, optionally upgrade Kubernetes to v1.25, v1.26, and v1.27.

An alternative path in the same scenario uses API Connect 10.0.5.1 on Kubernetes v1.22 as the interim release with the following upgrades:

1. On your API Connect 10.0.1.7 deployment, upgrade Kubernetes from v1.21 to v1.22.
2. Upgrade API Connect to 10.0.5.1 on Kubernetes v1.22.
3. On the API Connect 10.0.5.1 deployment, upgrade Kubernetes to v1.23, and then to v1.24.
4. Upgrade API Connect to 10.0.5.4 on Kubernetes v1.24.
5. On the API Connect 10.0.5.4 deployment, optionally upgrade Kubernetes to v1.25, v1.26, and v1.27.

Supported versions of OpenShift

The following tables provide a quick reference of the supported OpenShift versions for each API Connect release:

Table 2. API Connect and OpenShift Container Platform (OCP) compatibility matrix

API Connect version	OCP 4.6	OCP 4.7	OCP 4.8	OCP 4.9	OCP 4.10	OCP 4.12
10.0.1.7	✓	✓			✓	
10.0.1.8	✓	✓			✓	
10.0.1.9					✓	
10.0.1.11					✓	
10.0.1.12					✓	✓
10.0.1.15					✓	✓
10.0.4.0-ifix2 or later	✓	✓	✓	✓	✓	
10.0.5.0					✓	
10.0.5.1					✓	
10.0.5.2					✓	✓
10.0.5.3					✓	✓
10.0.5.4					✓	✓

Upgrading to an API Connect release that uses a newer version of OpenShift

When you upgrade API Connect, both your current deployment and your target release must support the same version of OpenShift. After the API Connect upgrade, you can optionally update OpenShift to the highest version supported by the new release of API Connect.

During the upgrade process, it is possible that API Connect will be temporarily running on an unsupported version of OpenShift because you must upgrade API Connect before upgrading OpenShift. For example, suppose that your current deployment is running API Connect 10.0.1.7 on OpenShift 4.6 and you want to upgrade to API Connect 10.0.5.4. The highest version of OpenShift supported with API Connect 10.0.1.7 is 4.6, but the lowest version of OpenShift supported by API Connect 10.0.5.4 is 4.10.

In this scenario, you must upgrade API Connect to the target release (10.0.5.4) even though it will temporarily run on OpenShift 4.6. Immediately after that upgrade is complete, you must upgrade OpenShift to at least version 4.10:

1. On your API Connect 10.0.1.7 deployment, leave OpenShift at version 4.6 and upgrade API Connect to 10.0.5.4.
2. On the API Connect 10.0.5.4 deployment, upgrade OpenShift to version 4.10 by completing all interim updates: 4.7, 4.8, 4.9, and then 4.10. You can optionally upgrade OpenShift to version 4.11 and then 4.12, which is the highest version supported by API Connect 10.0.5.4.

API Connect licenses

When you install a new version of API Connect, you must accept a license for the API Connect program that you purchased.

When you install or upgrade API Connect, the procedures include instructions for specifying your license. Be sure to specify a License ID from the version of API Connect that your installing, or are upgrading to.

API Connect 10.0.5.x licenses

For API Connect 10.0.5.3 and later, the licenses in Table 1 will be used as the master mod-level license for all Fix Pack and security roll-up releases in this stream. There are no changes to license terms, only updates to open-source notices documented in the notes for each.

Table 1. API Connect 10.0.5.3 and later licenses

License ID	Program Name	URL
L-VQYA-YNM22H	IBM API Connect Enterprise V10.0.5.3	https://ibm.biz/BdPGTL
L-SLCD-D9MBTA	IBM API Connect Enterprise Add-on for IBM Products V10.0.5.3	https://ibm.biz/BdPGTQ
L-XSJK-UMJP6P	IBM API Connect Professional V10.0.5.3	https://ibm.biz/BdPGTC

Table 2. API Connect 10.0.5.2 licenses

License ID	Program Name	URL
L-GVEN-GFUPVE	IBM API Connect Enterprise V10.0.5.2	https://ibm.biz/BdPPmt
L-BYRR-EEASJ3	IBM API Connect Enterprise Add-on for IBM Products V10.0.5.2	https://ibm.biz/BdPPm6
L-EMZJ-MQ4M24	IBM API Connect Professional V10.0.5.2	https://ibm.biz/BdPPmU
L-RJON-CJR3A7	IBM Cloud Pak for Integration - API Calls 2022.4.1	https://ibm.biz/BdPP5e

Table 3. API Connect 10.0.5.0 and 10.0.5.1 licenses

License ID	Program Name	URL
L-RJON-CEBLEH	IBM API Connect Enterprise PVU V10.0.5 If you purchased Cloud Pak for Integration, accept the API Connect Enterprise license when you purchase your Virtual Processor Cores.	https://ibm.biz/BdPFpN
L-RJON-CD3JHD	IBM Cloud Pak for Integration - API Calls 2022.2.1 If you purchased Cloud Pak for Integration - API Calls (which is an add-on product), select this license to track usage by the monthly number of API calls instead of the number of installed Virtual Processor Cores.	https://ibm.biz/BdPEBb
L-RJON-CEBLCF	IBM API Connect Enterprise PVU Add-on for IBM Products V10.0.5	https://ibm.biz/BdPFpW
L-RJON-CEBL97	IBM API Connect Professional PVU V10.0.5	https://ibm.biz/BdPF8B

Deployment procedures

You can use these procedures to deploy API Connect on either native Kubernetes or OpenShift.

- [Deploying on Kubernetes](#)
Use the procedures in this section to install API Connect on native Kubernetes distributions.
- [Deploying on OpenShift and Cloud Pak for Integration](#)
You can install API Connect in an OpenShift environment or as the API Management capability in IBM Cloud Pak for Integration.

Deploying on Kubernetes

Use the procedures in this section to install API Connect on native Kubernetes distributions.

- [Obtaining product files](#)
Obtain the product files, upload the images to a Docker registry, and decompress the operators and templates.
- [Deploying operators and cert-manager](#)
Deploy the Kubernetes operator files and install cert-manager.
- [Installing the API Connect subsystems](#)
Deploy the API Connect subsystem custom resources.
- [Installing a two data center deployment on Kubernetes](#)
Additional installation instructions for a two data center disaster recovery (2DCDR) deployment on Kubernetes.

Obtaining product files

Obtain the product files, upload the images to a Docker registry, and decompress the operators and templates.

Before you begin

- Ensure you have supported hardware and software. See [IBM API Connect Version 10 software product compatibility requirements](#).
- Complete the [Deployment requirements](#).
- Install and run Docker on the local machine being used for API Connect installation. Log in to your image registry.

About this task

From the [IBM Fix Central](#) site, download the Docker image-tool file of the API Connect subsystems. Next, you will upload the image-tool file to your Docker local registry. If necessary, you can populate a remote container registry with repositories. Then you can push the images from the local registry to the remote registry.

You will also download the Kubernetes operators, API Connect Custom Resource (CR) templates, and Certificate Manager, for use during deployment configuration.

Note:

- If you plan to migrate from Version 5 to Version 10, review [Migrating a Version 5 deployment](#) before you start the installation process.
- If you have API Connect installed and want to upgrade your release version, follow the upgrade instructions: [Upgrading API Connect in a Kubernetes runtime environment](#)

Procedure

1. Obtain the API Connect files:
 - a. Go to the [What's New in the latest version](#) information page.
 - b. Locate the Note: *You can access the latest files from <URL link>*. Select the <URL link> to go directly to the Announce page on Fix Central, where you can download files for the *latest version* of API Connect.

The following files are used for initial deployment on native Kubernetes:

IBM® API Connect <version> for Containers

Docker images for all API Connect subsystems

IBM® API Connect <version> Operator Release Files for Containers

Kubernetes operators and API Connect Custom Resource (CR) templates

IBM® API Connect <version> Toolkit for <operating_system_type>

Toolkit command line utility. Packaged standalone, or with API Designer or Loopback:

- IBM® API Connect <version> Toolkit for <operating_system_type>
- IBM® API Connect <version> Toolkit with Loopback for <operating_system_type>
- IBM® API Connect <version> Toolkit Designer with Loopback for <operating_system_type>

Not required during initial installation. After installation, you can download directly from the Cloud Manager UI and API Manager UI. See [Installing the toolkit](#).

IBM® API Connect <version> Local Test Environment

Optional test environment. See [Testing an API with the Local Test Environment](#)

IBM® API Connect <version> Security Signature Bundle File

Checksum files that you can use to verify the integrity of your downloads.

2. Optionally, you can verify the integrity of the downloaded files to ensure that they originated from IBM and are not modified. See [Signature verification by using PGP](#).

3. Load the image-tool image in your Docker local registry. The image is contained in the IBM® API Connect <version> for Containers download file. For example:

```
docker load < apiconnect-image-tool-<version>.tar.gz
```

Ensure that the registry has sufficient disk space for the files.

4. If your Docker registry requires repositories to be created before images can be pushed, create the repositories for each of the images listed by the image tool. (If your Docker registry does not require creation of repositories, skip this step and go to Step 5.)

a. Run the following command to get a list of the images from image-tool:

```
docker run --rm apiconnect-image-tool-<version> version --images
```

b. From the output of each entry of the form <image-name>:<image-tag>, use your Docker registry repository creation command to create a repository for <image-name>.

For example in the case of AWS ECR the command would be for each <image-name>:

```
aws ecr create-repository --repository-name <image-name>
```

5. Upload the image:

- If you do not need to authenticate with the docker registry, use:

```
docker run --rm apiconnect-image-tool-<version> upload <registry-url>
```

- Otherwise, if your docker registry accepts authentication with username and password arguments, use:

```
docker run --rm apiconnect-image-tool-<version> upload <registry-url> --username <username> --password <password>
```

- Otherwise, such as with IBM Container Registry, if you need the image-tool to use your local Docker credentials, first authenticate with your Docker registry, then upload images with the command:

```
docker run --rm -v ~/.docker:/root/.docker --user 0 apiconnect-image-tool-<version> upload <registry-url>
```

Note: The previous command does not work on macOS if Docker is configured to use the `osxkeychain` credential store. In this case, complete the following steps:

- Disable **Docker > Preferences... > Securely store Docker logins in the macOS keychain**.
- Inspect `~/.docker/config.json` to make sure that it does not contain "`credStore`": "`osxkeychain`", as some versions of Docker-for-mac may handle the setting correctly per <https://github.com/docker/for-mac/issues/4192>.
- Authenticate with the Docker registry you intend to upload to.
- Run the command:

```
docker run --rm -v ~/.docker:/root/.docker --user 0 apiconnect-image-tool-<version> upload <registry-url>
```

- Once the upload of images is successful you may enable **Docker > Preferences... > Securely store Docker logins in the macOS keychain**.

Docker authentication notes:

- Both HTTPS and HTTP are supported. Best practice for Docker registry security is to use HTTPS by utilizing standard Docker load tools, to ensure your images and platform are protected. However, when necessary you can use HTTP by specifying the `--tls-verify=false` flag.
- When using a Docker registry on `localhost`, you might encounter that the image-tool Docker container is on the Docker network and typically without access to `localhost`. You can workaround this issue by using the `--network host` argument to the Docker command. Note that `--network host` is a Docker argument, not an image-tool argument.
- Example of using `--network host` and `--tls-verify=false`:

```
$ docker run -d -p 5000:5000 --name registry registry:2
...
$ docker run --rm --network host apiconnect-image-tool-<version> upload localhost:5000 --tls-verify=false
...
```

Providing a certificate for verification, or disabling TLS verification

See the sample output of the tool usage for options to provide a certificate for verification or to disable TLS verification:

```
$ docker run --rm apiconnect-image-tool-<version> upload --help
upload docker images
```

Usage:

```
image-tool upload REGISTRY [flags]
```

Flags:

```
--cert-dir string  Directory with destination registry certificate tls.crt file
--username string  User name
-h, --help         help for upload
--password         password for <username>
--tls-verify       Verify TLS on destination registry (default true)
```

Global Flags:

```
--accept-license  Accept the license for API Connect
--debug           Enable debug logging
```

Notes:

- `--username=<username>` and `--password=<password>` can be used to specify credentials for authentication with the destination Docker registry.
- `--tls-verify=false` can be used to disable verification of the destination Docker registry certificate
- `--cert-dir <path>` can be used to provide a `tls.crt` file to be used for validation of the destination Docker registry certificate. For example:

```
docker run --rm -v <path-to-folder-with-tls.crt-file>:/cert apiconnect-image-tool-<version> upload <registry> --cert-dir /cert
```

6. Download the file IBM® API Connect <version> Operator Release Files for Containers

a. Decompress the downloaded Operator Release Files for Containers

Contents:

- API Connect Operator operator custom resource definition (CRDs).
- API Connect Operator Deployment and required resources CRDs.
- API Connect Operator Deployment and required resources CRDs, for multiple-namespace installations.
- DataPower Gateway Operator operator custom CRDs.
- API Connect custom resource templates, and Certificate Manager

The Operator Release Files for Containers files will be used later in deployment instructions.

Note: The Operator Release files may include catalog and operator source for OpenShift. These files are not used when deploying on native Kubernetes.

b. To access the API Connect custom resource templates and Certificate Manager, decompress the archive `helper_files.zip`.

The zip file contains custom resource templates for:

- Deployment of each subsystem.
- Backup and restore of each subsystem
- Custom certificates, both external and internal, for either standard deployments or 2-site HA deployments
- Certificate support for multi-namespace deployments
- Multi-site secret generation in a two data center deployment on Kubernetes.
- Ingress Issuer and Subsystem Certificates Resources, for either standard deployments or 2-site HA deployments
- Setting an administrator secret for DataPower Gateway

The templates will be used later in deployment instructions.

What to do next

Continue with [Deploying operators and cert-manager](#).

- **Signature verification by using PGP**

You can verify the integrity of files to ensure that they originated from IBM and are not modified.

Signature verification by using PGP

You can verify the integrity of files to ensure that they originated from IBM and are not modified.

About this task

You can use code signatures to verify that a downloaded file was created by IBM and that no bits in the file were changed. The signature files that are used are `*.asc`. The key/cert file is `PRD0003216key.pub.pgp`.

Procedure

1. Import the public key.

```
$ gpg --import PRD0003216key.pub.pgp
gpg: key 020ED6B5DBE65F3B: public key "APIConnect <psirt@us.ibm.com>" imported
gpg: Total number processed: 1
gpg:          imported: 1

$ gpg --list-key --fingerprint APIConnect
pub   rsa4096 2023-02-28 [SCE]
      39ED 1634 5FCD EDB5 D6ED  E860 020E D6B5 DBE6 5F3B
uid          [ unknown] APIConnect <psirt@us.ibm.com>
```

When the key is imported, it does not need to be imported again. It is not signed, but you should make a note of the fingerprint so you can compare to make sure that the key is IBM's:

```
39ED 1634 5FCD EDB5 D6ED  E860 020E D6B5 DBE6 5F3B
```

2. Verify the files.

```
$ $ gpg --verify helper_files.zip.asc
gpg: assuming signed data in 'helper_files.zip'
gpg: Signature made Mon 28 Sep 17:35:17 2020 PDT
gpg:          using RSA key 39ED16345FCDEDB5D6EDE860020ED6B5DBE65F3B
gpg: Good signature from "APIConnect <psirt@us.ibm.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 39ED 1634 5FCD EDB5 D6ED  E860 020E D6B5 DBE6 5F3B
```

The verification shows that the key matches the signature and that shows the fingerprint of the key, which should match the import:

```
39ED 1634 5FCD EDB5 D6ED  E860 020E D6B5 DBE6 5F3B
```

Deploying operators and cert-manager

Deploy the Kubernetes operator files and install cert-manager.

Before you begin

You must have completed the following installation tasks before you deploy the operators:

1. Review the installation requirements. See [Deployment requirements](#).
2. Obtain the API Connect product files. See [Obtaining product files](#).

About this task

Table 1 lists the operator that applies to each release of API Connect.

Table 1. Operators and operands for API Connect

API Connect release	API Connect Operator version	DataPower Operator version
10.0.5.4	3.4.0	1.6.8
10.0.5.3	3.3.0	1.6.6
10.0.5.2	3.2.0	1.6.3
10.0.5.1	3.1.0	1.6.2
10.0.5.0	3.0.0	1.6.0

The deployment procedures differ depending on whether your deployment cluster uses a single namespace or multiple namespaces. Follow the instructions that apply to your deployment cluster:

- [Deploying operators in a single-namespace API Connect cluster](#)
Deploy the Kubernetes operator files in a single namespace cluster.
- [Deploying operators in a multi-namespace API Connect cluster](#)
Deploy the Kubernetes operator files in multi-namespace cluster.
- [Installing cert-manager and certificates in a two data center deployment](#)
Additional instructions explaining how to install a cert-manager and certificates in a two data center disaster recovery deployment on Kubernetes.

Deploying operators in a single-namespace API Connect cluster

Deploy the Kubernetes operator files in a single namespace cluster.

Before you begin

- Ensure you completed the prerequisite tasks for installing operators. See [Deploying operators and cert-manager](#).
- Be sure to review your strategy for using certificates with API Connect. See [Deployment requirements](#).

Note: If you are deploying a multi-namespace API Connect cluster, do not use these instructions. Instead, go to [Deploying operators in a multi-namespace API Connect cluster](#).

Procedure

1. Ensure `KUBECONFIG` is set for the target cluster:

```
export KUBECONFIG=<path_to_cluster_config_YAML_file>
```

An example path is `/Users/user/.kube/clusters/<cluster_name>/kube-config-<cluster_name>.yaml`

2. Decide on a namespace to be used for installation.

If not using the default namespace, create the namespace in the Kubernetes cluster where the API Connect deployment will take place. Use the following command, replacing `<namespace>` with the namespace for the deployment:

```
kubectl create ns <namespace>
```

3. Install a cert-manager.

Use of a certificate manager adds convenience to the generation and management of certificate, but is not required. Whenever a custom resource (CR) takes a certificate secret name as input, you can point to any secret name, as long as the secret exists before deploying the CR, and the secret contains relevant certificate data. Typically, this is `tls.crt`, `tls.key`, and `ca.crt` files. See [Certificates in a Kubernetes environment](#).

Note:

Do not use this step to install a cert-manager if:

- You want to use custom certificates. See [Custom certificates on Kubernetes](#)
- You are deploying a two data center disaster recovery deployment on Kubernetes, see [Installing cert-manager and certificates in a two data center deployment](#).

After you complete one of the alternate sets of instructions in the links in this Note, continue with Step 4.

- a. Obtain the certificate manager.

API Connect v10 uses cert-manager v1.9.1 of cert-manager, which is a native [Kubernetes](#) certificate management controller.

You can obtain cert-manager v1.9.1 from the API Connect v10 distribution `helper_files.zip` archive, or download it from <https://github.com/cert-manager/cert-manager/releases/tag/v1.9.1>

Note: cert-manager is bundled as a common service in IBM CloudPak for Integration.

- b. Apply the CR:

```
kubectl apply -f cert-manager-1.9.1.yaml
```

Do not specify a custom namespace.

See <https://cert-manager.io/docs/release-notes/release-notes-1.9/>.

c. Wait for `cert-manager` pods to enter `Running 1/1` status before proceeding. To check the status:

```
kubectl get po -n cert-manager
```

There are 3 `cert-manager` pods in total.

4. Create a registry secret with credentials to be used to pull down product images.

Use the following command, replacing `<namespace>` with the namespace for your deployment, and replacing `<registry_server>` with the location of the Docker Registry where the installation product images were pushed:

```
kubectl create secret docker-registry apic-registry-secret
--docker-server=<registry_server>
--docker-username=<username@example.com> --docker-password=<password> --docker-email=<username@example.com> -n <namespace>
```

5. Create a registry secret for the DataPower registry with the credentials to be used to pull down product images.

Use the following command, replacing `<namespace>` with the desired namespace for the deployment:

```
kubectl create secret docker-registry datapower-docker-local-cred
--docker-server=<docker_server> --docker-username=<username@example.com> --docker-password=<password> --docker-email=
<username@example.com> -n <namespace>
```

6. Create a DataPower admin secret.

Use the following command, replacing `<namespace>` with the desired namespace for the deployment:

```
kubectl create secret generic datapower-admin-credentials --from-literal=password=admin -n <namespace>
```

The admin secret will be used for `$ADMIN_USER_SECRET` when deploying the gateway CR.

7. If you are using a namespace other than `default`, open `ibm-apiconnect.yaml` in a text editor. Replace each occurrence of `default` with the namespace for your deployment.

8. Open `ibm-apiconnect.yaml` in a text editor. Replace the value of each `image:` key with the location of the apiconnect operator images (from the `ibm-apiconnect` container and the `ibm-apiconnect-init` container), either uploaded to your own registry or pulled from a public registry.

```
serviceAccountName: ibm-apiconnect
imagePullSecrets:
- name: apic-registry-secret
initContainers:
- name: ibm-apiconnect-init
  image: <My_registry_or_public_registry>/ibm-apiconnect-operator-
init@sha256:0d3bbac7c67372ad013407a8049c69dbd08b1559b59a7606fcd87eb0f4519ce1
...
containers:
- name: ibm-apiconnect
  image: <My_registry_or_public_registry>/ibm-apiconnect-
operator@sha256:a7473ee26c252fca4931b682cd95d73b4149e4b5b773834408f9f22d0246a76c
```

9. If DataPower Gateway is desired in this installation, and if you are using a namespace other than `default`, open `ibm-datapower.yaml` in a text editor. Replace each occurrence of `default` with the namespace for your deployment.

10. Open `ibm-datapower.yaml` in a text editor.

a. Locate the `image:` key in the `containers` section of the deployment `yml` immediately after `imagePullSecrets:`. Replace the value of the `image:` key with the location of the `datapower` operator image, either uploaded to your own registry or pulled from a public registry.

b. Update the `IBM_DOCKER_HUB` and `IBM_ENTITLED_REGISTRY` value in the `env` section with the registry address where the images were uploaded with the `image-tool` in [Obtaining product files](#).

For example, the default entries are:

```
- name: IBM_ENTITLED_REGISTRY
  value: "cp.icr.io/cp/datapower"
- name: IBM_DOCKER_HUB
  value: "cp.icr.io/cp/datapower"
```

11. Install the `ibm-apiconnect` CRDs.

```
kubectl apply -f ibm-apiconnect-crds.yaml
```

12. Install the `ibm-apiconnect` Kubernetes deployment.

```
kubectl apply -f ibm-apiconnect.yaml
```

13. If DataPower Gateway is desired in this installation, install the `ibm-datapower` Kubernetes deployment for DataPower Gateway, with the following command, replacing `<namespace>` with the desired namespace for the deployment (`-n <namespace>` may be omitted if default namespace is being used for installation):

```
kubectl apply -f ibm-datapower.yaml -n <namespace>
```

Note: There is a known issue on Kubernetes version 1.19.4 or higher that can cause the DataPower operator to fail to start. In this case, the DataPower Operator pods can fail to schedule, and will display the status message: `no nodes match pod topology spread constraints (missing required label)`. For example:

```
0/15 nodes are available: 12 node(s) didn't match pod topology spread constraints (missing required label),
3 node(s) had taint {node-role.kubernetes.io/master: }, that the pod didn't tolerate.
```

You can workaround the issue by editing the DataPower operator deployment and re-applying it, as follows:

a. Delete the DataPower operator deployment, if deployed already:

```
kubectl delete -f ibm-datapower.yaml -n <namespace>
```


b. Open `ibm-datapower.yaml`, and locate the `topologySpreadConstraints:` section. For example:

```
topologySpreadConstraints:
- maxSkew: 1
  topologyKey: zone
  whenUnsatisfiable: DoNotSchedule
```

c. Replace the values for `topologyKey:` and `whenUnsatisfiable:` with the corrected values shown in the example below:

```
topologySpreadConstraints:
- maxSkew: 1
  topologyKey: topology.kubernetes.io/zone
  whenUnsatisfiable: ScheduleAnyway
```

d. Save `ibm-datapower.yaml` and deploy the file to the cluster:

```
kubectl apply -f ibm-datapower.yaml -n <namespace>
```

14. Install the `ingress-ca` Issuer to be used by `cert-manager`.

Note:

If you are installing a two data center disaster recovery deployment, skip this step, and follow the steps in: [Installing cert-manager and certificates in a two data center deployment](#).

a. Use the following command, replacing `<namespace>` with the desired namespace for the deployment:

```
kubectl apply -f ingress-issuer-v1.yaml -n <namespace>
```

The file `ingress-issuer-v1.yaml` is included in the release files contained in `helper_files.zip`

b. Validate that the command succeeded:

```
kubectl get certificates -n <namespace>
```

Example output indicating success:

NAME	READY	SECRET	AGE
analytics-ingestion-client	True	analytics-ingestion-client	70s
gateway-peering	True	gateway-peering	69s
gateway-service	True	gateway-service	69s
ingress-ca	True	ingress-ca	71s
portal-admin-client	True	portal-admin-client	71s
portal-tunnel-client	True	portal-tunnel-client	70s

The list of certificates might vary based on your deployment.

15. Continue with [Installing the API Connect subsystems](#).

Deploying operators in a multi-namespace API Connect cluster

Deploy the Kubernetes operator files in multi-namespace cluster.

Before you begin

- Ensure you completed the prerequisite tasks for installing operators. See [Deploying operators and cert-manager](#).
- Be sure to review your strategy for using certificates with API Connect. See [Deployment requirements](#).

About this task

Note: If you are deploying a single-namespace API Connect cluster, do not use these instructions. Instead, go to [Deploying operators in a single-namespace API Connect cluster](#).

- These instructions apply only to native k8s deployments. They do not apply to OpenShift deployments.
- The apiconnect operator is deployed as a cluster-scoped operator which watches every namespace in the cluster. The apiconnect operator can be installed in any namespace, although it is recommended that the API Connect operator is installed in its own dedicated namespace.
- The DataPower operator must be deployed in a namespace where gateway subsystem install is planned.
- Single ingress-ca certificate must be installed and used across the various namespaces where subsystems will be installed.

Procedure

1. Prepare your environment:

a. Ensure `KUBECONFIG` is set for the target cluster:

```
export KUBECONFIG=<path_to_cluster_config_YAML_file>
```

An example path is `/Users/user/.kube/clusters/<cluster_name>/kube-config-<cluster_name>.yaml`

b. Create namespaces on which subsystems are planned to be installed.

The required namespaces are:

- Apiconnect operator namespace
- Gateway subsystem namespace
- Management subsystem namespace
- Portal subsystem namespace

- Analytics subsystem namespace

The following example values and deployment are used throughout these instructions:

- An apiconnect operator is installed in `operator` namespace
- A DataPower operator and gateway subsystem is installed in `gtw` namespace
- The Management subsystem is installed in `mgmt` namespace
- The Portal subsystem is installed in `portal` namespace
- The Analytics subsystem is installed in `a7s` namespace

c. Multi-namespace templates are provided in `helper_files.zip`, inside the `release_files.zip` you downloaded in [Obtaining product files](#):

```
helper_files_unpack/multi-ns-support/
```

2. Install cert-manager and configure certificates:

Use of a certificate manager adds convenience to the generation and management of certificate, but is not required. Whenever a custom resource (CR) takes a certificate secret name as input, you can point to any secret name, as long as the secret exists before deploying the CR, and the secret contains relevant certificate data. Typically, this is `tls.crt`, `tls.key`, and `ca.crt` files. See [Certificates in a Kubernetes environment](#).

You can obtain cert-manager v1.9.1 from the API Connect v10 distribution `helper_files.zip` archive, or download it from <https://github.com/cert-manager/cert-manager/releases/tag/v1.9.1>.

a. Install cert-manager v1.9.1, do not specify a custom namespace as cert manager will be installed in its own namespace.

Note:

Do not use Step [2.a](#) to install a cert-manager if:

- You want to use custom certificates. See [Custom certificates on Kubernetes](#)
- You are deploying a two data center disaster recovery deployment on Kubernetes, see [Installing cert-manager and certificates in a two data center deployment](#)

i. `kubectl apply -f cert-manager-1.9.1.yaml`

ii. Wait for cert-manager pods to enter `Running 1/1` status before proceeding. Use the following command to check:

```
kubectl -n cert-manager get po
```

There are 3 cert-manager pods in total.

b. Install `ingress-ca` certificate:

- Locate `ingress-ca` certificate at `helper_files/multi-ns-support/ingress-ca-cert.yaml`
- Create `ingress-ca` certificate in one of the subsystems. For example, if we choose `mgmt` namespace:

```
kubectl -n mgmt apply -f ingress-ca-cert.yaml
```

c. Install common issuers on all the namespaces:

- Locate `common-issuer.yaml` in `helper_files/multi-ns-support/`
- Create `common-issuer.yaml` in all namespaces:

```
kubectl create -f common-issuer.yaml -n mgmt
kubectl create -f common-issuer.yaml -n gtw
kubectl create -f common-issuer.yaml -n ptl
kubectl create -f common-issuer.yaml -n a7s
```

d. Obtain the `ingress-ca` secret created by cert manager and create the secret on all namespaces:

Note:

The instructions in this step result in having a `selfsigning-issuer` in every namespace. This issuer is typically referenced by an API Connect CR in the namespace with:

```
...
microServiceSecurity: certManager
certManagerIssuer:
  name: selfsigning-issuer
  kind: Issuer
...
```

This issuer (and associated root certificate) does not need to be the same in every namespace, nor does it need to be the same for 2 CRs in the same namespace. Each subsystem can use its own `selfsigning-issuer`, because it is only used for certificates that are internal to the subsystem.

To ensure that the `ingress-issuer` is in sync across namespaces, it should be backed by the same `ingress-ca` certificate. This enables, for example, APIM to use a client certificate that matches the CA of the portal-admin ingress endpoint. The sync is achieved by copying around the `ingress-ca` certificate in each namespace, as shown in the following steps. It does not matter that it is the same issuer, or whether it has a different name. However it is important is that the `ingress-ca` is the same for all the "ingress-issuer" that are going to be used by the subsystems.

i. Make sure `ingress-ca` certificate `READY` status is set to true:

```
kubectl get certificate ingress-ca -n mgmt
NAME      READY  SECRET  AGE
ingress-ca  True   ingress-ca  9m24s
```

ii. Cert-manager creates a secret called `ingress-ca` in `mgmt` namespace which represents the certificate we created in Step [2.c](#).

iii. Use the following command to obtain the yaml format of the secret:

```
kubectl get secret ingress-ca -n <namespace-where-ingress-ca-cert-is-created> -o yaml > ingress-ca-secret-in-cluster.yaml
```

iv. Remove unwanted content from the secret yaml:

```
cat ingress-ca-secret-in-cluster.yaml | grep -v 'creationTimestamp' | grep -v 'namespace' | grep -v 'uid' | grep -v 'resourceVersion' > ingress-ca-secret.yaml
```

- v. Create the secret using the yaml we prepared in previous step on rest of the subsystem namespaces. In this example, in the `gtw`, `a7s` and `ptl` namespaces.

```
kubectl create -f ingress-ca-secret.yaml -n gtw
kubectl create -f ingress-ca-secret.yaml -n ptl
kubectl create -f ingress-ca-secret.yaml -n a7s
```

- e. Install management certs in management namespace:

- Locate `management-certs.yaml` in `helper_files/multi-ns-support/`
- Create `management-certs.yaml` in the `mgmt` namespace:

```
kubectl create -f management-certs.yaml -n mgmt
```

- f. Install gateway certs in the gateway namespace:

- Locate `gateway-certs.yaml` in `helper_files/multi-ns-support/`
- Create `gateway-certs.yaml` in the `gtw` namespace:

```
kubectl create -f gateway-certs.yaml -n gtw
```

3. Install the apiconnect operator:

- a. Install the `ibm-apiconnect` CRDs with the following command. Do not specify a namespace:

```
kubectl apply -f ibm-apiconnect-crds.yaml
```

- b. Create a registry secret with the credentials to be used to pull down product images with the following command, replacing `<namespace>` with the desired namespace for the apiconnect operator deployment.

```
kubectl -n <namespace> create secret docker-registry apic-registry-secret
--docker-server=<registry_server>
--docker-username=<username@example.com> --docker-password=<password>
--docker-email=<username@example.com>
```

- For example, replace `<namespace>` with `operator`.
- `docker-password` can be your artifactory API key.
- `-n <namespace>` may be omitted if `default` namespace is being used for installation

- c. Locate and open `ibm-apiconnect-distributed.yaml` in a text editor of choice. Then, find and replace each occurrence of `$OPERATOR_NAMESPACE` with `<namespace>`, replacing `<namespace>` with the desired namespace for the deployment. In this example, the namespace is `operator`.

- d. Also in `ibm-apiconnect-distributed.yaml`, locate the `image`: keys in the containers sections of the deployment yaml right below `imagePullSecrets`:. Replace the placeholder values `REPLACE-DOCKER-REGISTRY` of the `image`: keys with the docker registry host location of the apiconnect operator image (either uploaded to own registry or pulled from public registry).

- e. Install `ibm-apiconnect-distributed.yaml` with the following command

```
kubectl apply -f ibm-apiconnect-distributed.yaml
```

4. Install DataPower operator:

Deployment of the DataPower operator is only needed if you have at least one API Connect v10 Gateway subsystem (whether v5 compatible or not) to upgrade. If you are using DataPower in a different form factor such as an Appliance, you will not have an API Connect v10 Gateway subsystem to upgrade, and will not need the DataPower operator for your upgrade.

Note: The DataPower Operator supports multiple instances of DataPower in the same cluster, separated by namespace. When deploying into multiple namespaces, ensure that any cluster-scoped resources are created with unique names for separate installations, such that they are not overwritten by subsequent installations. For example, DataPower cluster-scoped resources include `ClusterRoleBindings`.

- a. Create a registry secret for the DataPower registry with the credentials to be used to pull down product images with the following command, replacing `<namespace>` with the desired namespace for the deployment .(In this case `gtw`):

```
kubectl -n <namespace> create secret docker-registry datapower-docker-local-cred
--docker-server=<registry_server>
--docker-username=<username@example.com> --docker-password=<password>
--docker-email=<username@example.com>
```

- For example, replace `<namespace>` with `gtw`.
- `docker-password` can be your artifactory API key.
- `-n <namespace>` may be omitted if `default` namespace is being used for installation

- b. Create a DataPower admin secret, replacing `<namespace>` with the desired namespace for the deployment. For example, `gtw`. This secret will be used for `$ADMIN_USER_SECRET` later in the Gateway CR:

```
kubectl -n <namespace> create secret generic datapower-admin-credentials --from-literal=password=admin
-n
```

`<namespace>` may be omitted if `default` namespace is being used for installation.

- c. Locate and open `ibm-datapower.yaml` in a text editor of choice. Then, find and replace each occurrence of `default` with `<namespace>`, replacing `<namespace>` with the desired namespace for the deployment. For example, `gtw`.

- d. Install `ibm-datapower.yaml` with the following command:

```
kubectl -n <namespace> apply -f ibm-datapower.yaml
```

Note: There is a known issue on Kubernetes version 1.19.4 or higher that can cause the DataPower operator to fail to start. In this case, the DataPower Operator pods can fail to schedule, and will display the status message: `no nodes match pod topology spread constraints (missing required label)`. For example:

```
0/15 nodes are available: 12 node(s) didn't match pod topology spread constraints (missing required label),
3 node(s) had taint {node-role.kubernetes.io/master: }, that the pod didn't tolerate.
```

You can workaround the issue by editing the DataPower operator deployment and re-applying it, as follows:

- Delete the DataPower operator deployment, if deployed already:

```
kubectl delete -f ibm-datapower.yaml -n <namespace>
```

- ii. Open `ibm-datapower.yaml`, and locate the `topologySpreadConstraints:` section. For example:

```
topologySpreadConstraints:
- maxSkew: 1
  topologyKey: zone
  whenUnsatisfiable: DoNotSchedule
```

- iii. Replace the values for `topologyKey:` and `whenUnsatisfiable:` with the corrected values shown in the example below:

```
topologySpreadConstraints:
- maxSkew: 1
  topologyKey: topology.kubernetes.io/zone
  whenUnsatisfiable: ScheduleAnyway
```

- iv. Save `ibm-datapower.yaml` and deploy the file to the cluster:

```
kubectl apply -f ibm-datapower.yaml -n <namespace>
```

5. Next, install the subsystems. Continue with [Installing the API Connect subsystems](#).

Note: Unless otherwise stated, when performing the individual subsystem upgrades the `<namespace>` value of example `kubectl` commands for a given subsystem should be set to the namespace for the particular subsystem component they are acting on.

Installing cert-manager and certificates in a two data center deployment

Additional instructions explaining how to install a cert-manager and certificates in a two data center disaster recovery deployment on Kubernetes.

Before you begin

Before you install a cert-manager, you should know your strategy for using certificates with API Connect. Review the certificate requirements in [Deployment requirements](#).

You should also have already completed [Obtaining product files](#).

About this task

You must create certificates and keys for API Manager and the Developer Portal on both data centers, dc1 and dc2, and ensure that they match on both.

Use these instructions to install the supplied cert-manager and `ingress-issuer-v1.yaml`. Use these instructions if you are *not* using custom certificates.

Note that `ingress-issuer-v1.yaml` is supplied by cert-manager.

Note:

Do not use these instructions if you want to use custom certificates. See [Custom certificates on Kubernetes](#).

Procedure

1. Install a cert-manager.

Use of a certificate manager adds convenience to the generation and management of certificate, but is not required. Whenever a custom resource (CR) takes a certificate secret name as input, you can point to any secret name, on condition that the secret exists before you deploy the CR, and that the secret contains relevant certificate data. Typically, this is `tls.crt`, `tls.key`, and `ca.crt` files. See [Certificates in a Kubernetes environment](#).

- a. Obtain the certificate manager.

API Connect v10 uses cert-manager v1.9.1 of cert-manager, which is a native [Kubernetes](#) certificate management controller.

You can obtain cert-manager v1.9.1 from the API Connect v10 distribution `helper_files.zip` archive, or download it from <https://github.com/cert-manager/cert-manager/releases/tag/v1.9.1>

Note: cert-manager is bundled as a common service in IBM CloudPack for Integration.

- b. Apply the CR:

```
kubectl apply -f cert-manager-1.9.1.yaml
```

Do not specify a custom namespace.

See <https://cert-manager.io/docs/release-notes/release-notes-1.5/>.

- c. Wait for `cert-manager` pods to enter `Running 1/1` status before proceeding. To check the status:

```
kubectl get po -n cert-manager
```

There are 3 `cert-manager` pods in total.

2. Use the following steps to allow `ingress-ca` secrets to be the same on both data centers.

- a. On DC1 apply the file `ingress-issuer-v1-dc1.yaml`:

```
kubectl -n <namespace> apply -f ingress-issuer-v1-dc1.yaml
```

- b. Validate that the command succeeded:

```
kubectl get certificates -n <namespace>
```

- c. Export `ingress-ca` secret as a yaml from DC1:

```
kubectl -n <namespace> get secret ingress-ca -o yaml > ingress-ca.yaml
```

- d. Edit the `ingress-ca.yaml` file to remove all `annotations`, `labels`, `creationTimestamp`, `managedFields`, `manager`, `operation`, `time`, `resourceVersion`, `selfLink`, and `uid`. Also, if you are using a different `namespace` in DC2, then update the `namespace` field.
- e. Copy the `ingress-ca.yaml` from DC1 to DC2 and apply that file on DC2:

```
kubectl -n <namespace> apply -f ingress-ca.yaml
```

- f. On DC2 apply the file `ingress-issuer-v1-dc2.yaml`:

```
kubectl -n <namespace> apply -f ingress-issuer-v1-dc2.yaml
```

- g. Use the following commands to test that they are the same, on DC1 run:

```
kubectl -n <namespace> get secrets ingress-ca -o yaml | grep tls.crt | grep -v 'f:tls' | awk '{print $2}' | base64 -d > /tmp/ingress.pem.dc1
```

- h. On DC2 run:

```
kubectl -n <namespace> get secrets ingress-ca -o yaml | grep tls.crt | grep -v 'f:tls' | awk '{print $2}' | base64 -d > /tmp/ingress.pem.dc2
```

- i. To see the differences run:

```
diff /tmp/ingress.pem.dc1 /tmp/ingress.pem.dc2
```

The files should be the same.

- j. On DC2, to ensure that the certificates are working correctly and that they are using the `ingress-ca` secret. First, get the `portal-admin-client.crt` file:

```
kubectl -n <namespace> get secrets portal-admin-client -o yaml | grep tls.crt | awk '{print $2}' | base64 -d > /tmp/admin-client.crt
```

- k. Test that it is working by using OpenSSL:

```
openssl verify -verbose -CAfile /tmp/ingress.pem.dc1 /tmp/admin-client.crt
```

If it is working, you should see:

```
/tmp/admin-client.crt: OK
```

3. Continue with [Deploying operators and cert-manager](#).

Installing the API Connect subsystems

Deploy the API Connect subsystem custom resources.

Before you begin

Attention: API Connect is not supported on a FIPS-enabled environment.

Before you install the subsystems, verify that you completed the following prerequisite tasks:

1. [Obtaining product files](#)
2. [Deploying operators and cert-manager](#)

About this task

Use the CR (custom resource) files to install the Management, Gateway, Analytics, and Portal subsystems for your API Connect cloud. CRs are extensions of the Kubernetes API. API Connect installs its subsystems using individual CRs. One CR is deployed per one managed subsystem. The API Connect operator manages the lifecycle of the subsystem CRs and deploys and manages the microservice for each one.

CR templates are provided in the `helper_files` archive to install each API Connect subsystem. You uncompressed these files into an installation folder of your choosing in [Obtaining product files](#).

- The API Connect operators and operands must be from the same version of API Connect. For example, the API Connect operator 10.0.4.0 does not successfully deploy the API Connect management subsystem (operand) from Version 10.0.3.0.
- You can deploy multiple instances of API Connect into the same cluster by using different namespaces.
- Deploying multiple similar subsystems in a single namespace is not supported. For example, deploying two management subsystems, `mgmt1`, `mgmt2` in a single namespace where `apiconnect` operator is installed.
- For more information on Custom Resources in Kubernetes see <https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/>.
- For details on how the API Connect operator manages the CRs using the Kubernetes operator pattern, see <https://kubernetes.io/docs/concepts/extend-kubernetes/operator/>.
- For guidance on choosing object names and IDs in Kubernetes, see <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>.

Procedure

Complete each of the following instructions:

1. [Installing the Management subsystem cluster](#).
2. [Installing the DataPower Gateway subsystem](#).
3. [Installing the Developer Portal subsystem](#).
4. [Installing the Analytics subsystem](#).
5. When all subsystems are running, verify you can access the API Connect Cloud Manager. Enter the Cloud Manager endpoint URL in your browser.

Tip:

You can find your Cloud Manager endpoint URL as follows:

The property `cloudManagerEndpoint` is specified in the management CR (`management_cr.yaml`). When you edited the template, you specified a value for `$STACK_HOST`:

```
cloudManagerEndpoint:
  annotations:
    cert-manager.io/issuer: ingress-issuer
  hosts:
  - name: admin.$STACK_HOST
    secret: cm-endpoint
```

The template URL is `https://admin.<STACK_HOST>/admin`. The `/admin` suffix at the end is the name of the Cloud Manager administrator.

For example, if `$STACK_HOST` is `myhost.subnet.example.com`, the Cloud Manager endpoint URL is:

```
https://admin.myhost.subnet.example.com/admin
```

The first time that you access the Cloud Manager user interface, you enter `admin` for the user name and `7iron-hide` for the password. You will be prompted to change the Cloud Administrator password and email address. See [Accessing the Cloud Manager user interface](#).

What to do next

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the [Cloud Manager configuration checklist](#).

- [Installing the Management subsystem cluster](#)
Install the Management subsystem cluster
- [Installing the DataPower Gateway subsystem](#)
Install the DataPower Gateway subsystem
- [Installing the Developer Portal subsystem](#)
Install the Developer Portal subsystem.
- [Installing the Analytics subsystem](#)
Install the IBM API Connect analytics subsystem.

Installing the Management subsystem cluster

Install the Management subsystem cluster

Before you begin

Before you start this task, you should have already:

- Reviewed the Kubernetes requirements in [Pre-installation requirements](#)
- Completed [Deploying operators and cert-manager](#)

About this task

Edit the custom resource template for the Management system, apply the resource, verify that the pods are up and running, and verify that you can connect to the API Connect Cloud Manager.

Procedure

1. Edit the `management_cr` template CR, to replace the placeholders with values for your deployment.

`$APP_PRODUCT_VERSION`

API Connect application version for the subsystems.

```
version: <version_number>
```

Example version number: 10.0.5.4

`$SECRET_NAME`

Use for image pull.

```
imagePullSecrets:
- apic-registry-secret
```

`$PROFILE`

Specify your Management subsystem profile, where `n` indicates number of replicas, `c` number of cores, and `m` is the minimum memory allocation in GB. For more information on profiles, see [Planning your deployment topology](#).

`$DOCKER_REGISTRY`

The host name of the Docker Registry to which you uploaded the installation images. For example:

```
my.docker.registry.domain.example.com.
```

`$INGRESS_CLASS`

The ingress class that you want the endpoint to use. This is an optional property and if not specified, the ingress class with annotation `ingressclass.kubernetes.io/is-default-class`:

`true` is used. If such an ingress class does not exist in the Kubernetes environment, then `nginx` is used. If you do set this value, it must refer to a valid ingress class configured in your Kubernetes system.

Note: This property is commented out in the template CR file. If you set this value, make sure to also uncomment it.

`STACK_HOST`

The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can do one of the following:

- Subdomain customization only
Accept the prefixes predefined for the ingress hostnames to use and just replace all instances of `STACK_HOST` to be the desired ingress subdomain for the API Connect stack. For example, if your host is `myhost.subnet.example.com`:

```
cloudManagerEndpoint:
  < ... >
  hosts:
  - name: admin.myhost.subnet.example.com
    secret: cm-endpoint

apiManagerEndpoint:
  < ... >
  hosts:
  - name: manager.myhost.subnet.example.com
    secret: apim-endpoint

platformAPIEndpoint:
  < ... >
  hosts:
  - name: api.myhost.subnet.example.com
    secret: api-endpoint

consumerAPIEndpoint:
  < ... >
  hosts:
  - name: consumer.myhost.subnet.example.com
    secret: consumer-endpoint
```

- Complete hostname customization

Change both the predefined prefixes and the `STACK_HOST` subdomain to match your desired hostnames.

For example, for `cloudManagerEndpoint`, you can replace `admin.STACK_HOST` with `my.cloudmgr.myhost.subnet.example.com`, where `my.cloudmgr` replaces `admin`, and `myhost.subnet.example.com` replaces `STACK_HOST`. For example:

```
cloudManagerEndpoint:
  < ... >
  hosts:
  - name: my.cloudmgr.myhost.subnet.example.com
    secret: cm-endpoint
```

You can do this for some or all of the hostnames, depending on your customization requirements.

`STORAGE_CLASS`

The Kubernetes storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: `kubectl get sc`.

```
storageClassName: local-storage
```

2. Edit the `license` setting:

- Set `accept`: to `true` to accept the license. Note that the default value is `false`. If you do not accept the license, the Operator will not install the subsystem.
- Set `metric`: to track your product usage. Enter the unit of measure that is used for your program license:
 - `PROCESSOR_VALUE_UNIT` - Default value. If you leave the field blank, this value is used.
 - `MONTHLY_API_CALL` - Applies only to the IBM API Connect Hybrid Entitlement program.

For information on tracking monthly call volume, see [Tracking API volume for auditing and compliance](#).

- Set `use`: to either `production` or `nonproduction`, to match the license you purchased.
- Set `license`: to the License ID for the version of API Connect that you purchased. See [API Connect licenses](#).

Example entry to accept the license for a production system:

```
license:
  accept: true
  metric: PROCESSOR_VALUE_UNIT
  use: production
  license: L-RJON-BZ5LSE
```

3. If installing with custom internal certificates, specify a site name.

- The site name is used as the identifier for the PostgreSQL database cluster used by API Connect. This name should already have been decided upon and used to update the `custom-certs-internal.yaml` file as described in [Generating custom certificates](#).
- You must now add your chosen site name to `management_cr.yaml`, by creating a `siteName` property anywhere inside the `spec` block, set to the chosen site name:

```
siteName: <site_name>
```

Replace `<site_name>` with your chosen site name.

4. It is recommended to allocate 100Gi for write-ahead logging (WAL) storage.

The default storage is:

- One replica profile - 30Gi
- Three replica profile - 47Gi

For best performance, edit the CR to add the following entries, and set `volumeSize`: to 100Gi:

```
spec:
  dbArchiveVolumeClaimTemplate:
    storageClassName: <storage-class>
    volumeSize: <volume-size>
```

5. Configure backups for the management subsystem.

Important: Note that in order to restore your deployment in the event of a disaster where you must redeploy a new cluster, you must take specific configuration actions **before** installing the Management subsystem. See [Preparing the management subsystem for disaster recovery](#).

You can review the backup and restore features here: [Configuring backup settings for fresh install of the Management subsystem](#)

6. Optional: If you are installing as part of a two data center disaster recovery set up, complete the steps in [Installing a two data center deployment on Kubernetes](#) before you apply the edited file.

7. Install the management Custom Resource, replacing <namespace> with the target installation namespace in the Kubernetes cluster.

```
kubectl apply -f management_cr.yaml -n <namespace>
```

8. Verify that the Management subsystem is fully installed:

```
kubectl get ManagementCluster -n <namespace>
```

The installation has completed when the **READY** status is **True**, and the **SUMMARY** reports that all services are online (e.g. 9/9). For example:

NAME	READY	SUMMARY	VERSION	RECONCILED	VERSION	AGE
management	True	16/16	<version>	<version-build>		7m17s

9. Retain the encryption secret and database credential secrets in a safe location, in case you need to restore during a disaster recovery scenario. Complete the steps in [Preparing the management subsystem for disaster recovery](#).

Important: If you do not complete the steps in [Preparing the management subsystem for disaster recovery](#), you will not be able to restore your management subsystem to a new cluster during a disaster recovery scenario.

10. Check your connection to the Cloud Manager user interface on the Management subsystem on your Cloud Manager endpoint.

For example, if you accepted the default prefix of **admin**, and you set **\$STACK_HOST** to **myhost.subnet.example.com**, the Cloud Manager endpoint URL is:

```
https://admin.myhost.subnet.example.com/admin
```

The first time that you access the Cloud Manager user interface, you enter **admin** for the user name and **7iron-hide** for the password. You will be prompted to change the Cloud Administrator password and email address. For further details, see [Accessing the Cloud Manager user interface](#).

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the [Cloud Manager configuration checklist](#).

Installing the DataPower Gateway subsystem

Install the DataPower Gateway subsystem

Before you begin

If you plan to install the Gateway subsystem in a remote cluster, be sure to satisfy the following requirements:

- Install both the API Connect and the DataPower operators.
The versions of the operators on the remote cluster and the version of the Gateway operand must exactly match the versions used in the main cluster.
- Make sure the remote cluster has the certificates:
 - If you are using certificate manager, ensure that the ingress-ca is synced between sites.
 - If you are not using certificate manager, copy the client gateway certificates to the remote cluster and reference them in the Gateway CR.

About this task

Note: An alternate deployment scenario is to use a physical DataPower appliance for the gateway. If you are using an appliance for the gateway, do not install using API Connect Kubernetes CRs, since you do not need to configure a gateway subsystem in your Kubernetes cluster when using an appliance. For an appliance-based gateway, you must configure two endpoints in DataPower to be used as the Gateway Service Management Endpoint and the API invocation Endpoint. Enter these endpoints in the Configure Gateway Service screen in Cloud Manager. See [Configuring DataPower Gateway for API Connect](#) for instructions for configuring a DataPower appliance.

The installation folder where the **helper_files.zip** was extracted contains two templates for Gateway service. The **v5cgateway_cr.yaml** template is for the v5 compatible Gateway, and the **apigateway_cr.yaml** template is for the API Gateway.

Procedure

1. Edit the template CR, either **apigateway_cr.yaml** or **v5cgateway_cr.yaml**, to replace the placeholders with values for your deployment.

```
$APP_PRODUCT_VERSION
API Connect application version for the subsystems.
```

```
version: <version_number>
```

Example version number: 10.0.5.4

```
$PROFILE
```


Specify your gateway profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see [Planning your deployment topology](#).

`$SECRET_NAME`

Use for image pull.

```
imagePullSecrets:
- apic-registry-secret
```

`$DOCKER_REGISTRY`

The host name of the Docker Registry to which you uploaded the installation images. For example:

```
my.docker.registry.domain.example.com.
```

`$INGRESS_CLASS`

The ingress class that you want the endpoint to use. This is an optional property and if not specified, the ingress class with annotation `ingressclass.kubernetes.io/is-default-class: true` is used. If such an ingress class does not exist in the Kubernetes environment, then `nginx` is used. If you do set this value, it must refer to a valid ingress class configured in your Kubernetes system.

Note: This property is commented out in the template CR file. If you set this value, make sure to also uncomment it.

`$STACK_HOST`

The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore " `_` " character. You can do one of the following:

- Subdomain customization only
Accept the prefixes predefined for the ingress hostnames to use and just replace all instances of `STACK_HOST` to be the desired ingress subdomain for the API Connect stack. For example, if your host is `myhost.subnet.example.com`:

```
gatewayEndpoint:
  < ... >
  hosts:
  - name: rgw.myhost.subnet.example.com
    secret: gwv6-endpoint

gatewayManagerEndpoint:
  < ... >
  hosts:
  - name: rgwd.myhost.subnet.example.com
    secret: gwv6-manager-endpoint
```

- Complete hostname customization
Change both the predefined prefixes and the `STACK_HOST` subdomain to match your desired hostnames.

For example, for `gatewayEndpoint`, you can replace `rgw.$STACK_HOST` with `my.gateway.endpoint.myhost.subnet.example.com`, where `my.gateway.endpoint` replaces `rgw`, and `myhost.subnet.example.com` replaces `$STACK_HOST`:

```
gatewayEndpoint:
  < ... >
  hosts:
  - name: my.gateway.endpoint.myhost.subnet.example.com
    secret: gwv6-endpoint
```

You can do this for some or all of the hostnames, depending on your customization requirements.

`$STORAGE_CLASS`

The Kubernetes storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: `kubectl get sc`.

```
storageClassName: local-storage
```

`$ADMIN_USER_SECRET`

Edit `$ADMIN_USER_SECRET` to set the value you created in [Deploying operators and cert-manager](#).

```
adminUserSecret: $ADMIN_USER_SECRET
```

`$PLATFORM_CA_SECRET`

Set to `ingress-ca`. The `$PLATFORM_CA_SECRET` contains the CA certificate for the platform REST API endpoint. The gateway makes calls to the platform REST API and this property must be set so that the gateway can verify the server certificate.

```
mgmtPlatformEndpointCASecret:
  secretName: $PLATFORM_CA_SECRET
```

2. Edit the `license` setting:

- Set `accept` to `true` to accept the license. Note that the default value is `false`. If you do not accept the license, the Operator will not install the subsystem.
- Set `metric` to track your product usage. Enter the unit of measure that is used for your program license:
 - `PROCESSOR_VALUE_UNIT` - Default value. If you leave the field blank, this value is used.
 - `MONTHLY_API_CALL` - Applies only to the IBM API Connect Hybrid Entitlement program.For information on tracking monthly call volume, see [Tracking API volume for auditing and compliance](#).
- Set `use` to either `production` or `nonproduction`, to match the license you purchased.
- Set `license` to the License ID for the version of API Connect that you purchased. See [API Connect licenses](#).

Example entry to accept the license for a production system:

```
license:
  accept: true
  metric: PROCESSOR_VALUE_UNIT
  use: production
  license: L-RJON-BZ5LSE
```

3. Optional: If applicable to your deployment, you can override the resource settings for CPU and memory.

- CPU override

When you specify `spec.license.use: nonproduction` and a one replica profile `profile: n1xc4.m8`, the default CPU request for the gateway pod is 1 CPU. You can manually override this setting. For example, to request 2 CPUs:

```
template:
- name: datapower
  containers:
  - name: gateway
    resources:
      requests:
        cpu: 2
```

- When you specify `spec.license.use: production` and a one replica profile `profile: n1xc4.m8`, the default CPU request for the gateway pod is 4 CPUs.

- The CPU limit cannot be specified separately from the CPU request. The value for the CPU request is used as both CPU request and CPU limit in the Gateway CR.

- Memory override

The default memory limit and request is 8Gi. The memory limit should always be greater than or equal to the memory request.

For example, to request 10Gi and set a limit of 12Gi:

```
template:
- name: datapower
  containers:
  - name: gateway
    resources:
      limits:
        memory: 12Gi
      requests:
        cpu: 2
        memory: 10Gi
```

Note: The override mechanism is specific only to the CPU and memory values, and cannot be used for other resource requests in the DataPower Gateway CR.

4. For `apigateway_cr.yaml` only, you can optionally enable `openTracing`.

a. Edit the CR to add the following properties, and supply values for each:

```
openTracing:
  enabled: false # if true provide below details
  odTracingRegistrationHostname: $OD_TRACING_REGISTRATION_HOSTNAME
  odTracingDataHostname: $OD_TRACING_HOSTNAME
  imageAgent: $AGENT_IMAGE
  imageCollector: $COLLECTOR_IMAGE
```

Where:

enabled

```
enabled: true
```

`$OD_TRACING_REGISTRATION_HOSTNAME`

```
odTracingRegistrationHostname: icp4i-od.tracing.svc
```

The hostname of the registration server for OpenTracing.

`$OD_TRACING_HOSTNAME`

```
odTracingDataHostname: od-store-od.tracing.svc
```

The hostname of the data server for OpenTracing.

`$AGENT_IMAGE`

```
imageAgent: icp4i_od_agent:1.0.0-amd64
```

Image will be pulled from the `imageRegistry` of this CR.

`$COLLECTOR_IMAGE`

```
imageCollector: icp4i_od_collector:1.0.0-amd64
```

Image will be pulled from the `imageRegistry` of this CR.

b. During the OpenTracing registration process, name the secret that stores credentials `icp4i-od-store-cred`, and add it to the cluster and into the installation namespace.

5. Enable or disable `syslogConfig`.

```
syslogConfig:
  enabled: false # if true, provide below details
  # remoteHost: $DATAPOWER_SYSLOG_TCP_REMOTE_HOST # must be a string
  # remotePort: $DATAPOWER_SYSLOG_TCP_REMOTE_PORT # must be an int
  # secretName: $DATAPOWER_SYSLOG_TCP_TLS_SECRET # must be a string
```

- To disable syslog, remove the `syslogConfig` block from `apigateway_cr.yaml` and `v5cgateway_cr.yaml`. Alternatively, you can set `syslogConfig.enabled` to `false`.

- To enable `syslogConfig` in the Gateways, set the following values in `apigateway_cr.yaml` and `v5cgateway_cr.yaml`:

enabled

```
enabled: true
```

`$DATAPOWER_SYSLOG_TCP_REMOTE_HOST`

```
remoteHost: <remote.host>
```

String. The hostname of the server for `syslogConfig`.

`$DATAPOWER_SYSLOG_TCP_REMOTE_PORT`

```
remotePort: <remote_port_number>
```

Integer. The port number of the server for `syslogConfig`.

```
$DATAPOWER_SYSLOG_TCP_TLS_SECRET
secretName: <secret>
String. The TLS secret name of the server for syslogConfig.
```

- Optional: If applicable to your deployment, update the GatewayCluster CR to configure `DataPowerService` APIs to customize your DataPower deployment. See [Customizing a DataPower deployment](#).
- To enable mutual TLS between the Management client and the Gateway service's manager endpoint, add `mtlsValidateClient` to the `spec` section:

```
spec:
  mtlsValidateClient: true
```

This ensures that the gateway service authenticates incoming requests from the API Manager, such as gateway service registration, and publishing APIs to the gateway service. Specifically, the gateway service requires that incoming requests present a certificate that was signed by the same CA that was used to sign the gateway service management endpoint. The gateway service management endpoint secret is specified under `gatewayManagerEndpoint.hosts.secretName`. The API Manager's gateway client's TLS credentials are specified in the ManagementCluster CR under `gateway.client.secretName`.
 Note: In releases previous to 10.0.5.3, this property is called `validateApimClient`.

- Optional: If you want to enable JWT security for communications between the management and gateway, add and set the property `jwtksUrl`.

```
spec:
  ...
  jwtksUrl: <JWTKS URL>
```

where `<JWTKS URL>` is the URL of the JWKS endpoint hosted on the management subsystems. To find out the `jwtksUrl`, describe the management CR and check the `status` section:

```
kubectl describe mgmt -n <namespace>
...
status:
- name: jwtksUrl
  secretName: api-endpoint
  type: API
  uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information on JWT security, see [Enable JWT security instead of mTLS](#).

DataPower version 10.5.0.4 only: To enable JWT on the gateway to analytics communications flow, you also must add a `dataPowerOverride` section to the gateway CR:

```
spec:
  jwtksUrl: <JWTKS URL>
  dataPowerOverride:
    image: customregistry.com/custom-image-datapower:10.5.0.5
    version: 10.5.0.5
    license: X-XXXX-XXXXXX
```

- Optional: If you want to use the [In-cluster](#) inter-subsystem communication feature, then uncomment the `mgmtPlatformEndpointSvcCASecret` section, and set the `secretName`:

```
mgmtPlatformEndpointSvcCASecret:
  secretName: management-ca # Usually management-ca
```

Unless you customized your internal certificates, this is always `management-ca`.

- Optional: Enable autoscaling of gateway pods to ensure high availability of DataPower pods. See [Enabling autoscaling of gateway pods](#).

- Install the Gateway Custom Resource by applying the Gateway template file:

For a DataPower API Gateway, run the following command:

```
kubectl apply -f apigateway_cr.yaml -n <namespace>
```

For a DataPower Gateway (v5 compatible), run the following command:

```
kubectl apply -f v5cgateway_cr.yaml -n <namespace>
```

- To verify that the Gateway subsystem(s) are fully installed, run the following command:

```
kubectl get GatewayCluster -n <namespace>
```

The installation has completed when the `READY` status is `True`, and the `SUMMARY` reports that all services are online (2/2) for all the Gateway subsystems that were installed. Example:

NAME	READY	SUMMARY	VERSION	RECONCILED VERSION	AGE
gww5	True	2/2	<version>	<version-build>	7m31s
gww6	True	2/2	<version>	<version-build>	7m32s

There is no need to wait for the `READY` status to be `True` before moving on to the next Subsystem installation.

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the [Cloud Manager configuration checklist](#).

Installing the Developer Portal subsystem

Install the Developer Portal subsystem.

About this task

Important:

- To enable effective high availability for your Portal service, you need a latency that is less than 50 ms between all portal-db pods to avoid the risk of performance degradation. Servers with uniform specifications are required, as any write actions occur at the speed of the slowest portal-db pod, as the write actions are synchronous across the cluster of portal-db pods. It is recommended that there are three servers in each cluster of portal-db pods for the high availability configuration. The three servers can be situated in the same availability zone, or across three availability zones to ensure the best availability. However, you can configure high availability with two availability zones.
- The Portal does not do certain operations if the free space on certain volumes falls beneath predefined limits. In particular:
 - The `databaseVolumeClaimTemplate` needs at least 4GB of free space to create new sites, restore site backups, or upgrade or change the URL of existing sites.
 - The `webVolumeClaimTemplate` needs at least 256MB free space to create new sites, restore site backups, or upgrade or change the URL of existing sites.
- Ensure that your kernel or Kubernetes node has the value of its `inotify` watches set high enough so that the Developer Portal can monitor and maintain the files for each Developer Portal site. If set too low, the Developer Portal containers might fail to start or go into a `non-ready` state when this limit is reached. If you have many Developer Portal sites, or if your sites contain a lot of content, for example, many custom modules and themes, then a larger number of `inotify` watches are required. You can start with a value of 65,000, but for large deployments, this value might need to go up as high as 1,000,000. The Developer Portal containers take `inotify` watches only when they need them. The full number is not reserved or held, so it is acceptable to set this value high.
- Ensure that your kernel or Kubernetes node has a value of `nproc` (maximum number of processes) that applies to the user ID of the Portal pods that have been assigned, and that it is high enough to allow all of the Portal processes to execute. For smaller installations this might be as low as 16384, but for larger installations that have more concurrent web calls, you might need as many as 125205. If the number is too low, you will see errors like `"fork: retry: Resource temporarily unavailable"` in the Portal logs. Note that this value might need to be even higher if other, non-Portal, pods are sharing the same user ID.
- Ensure that your kernel or Kubernetes node has a value of `nofiles` (maximum number of open file descriptors) that applies to the user ID of the Portal pods that have been assigned, and that it is high enough to allow the Portal to open all of the files that it requires. For smaller installations this might be as low as 16384, but for larger installations that have more concurrent web calls and Portal web sites, you might need as many as 1048576. If the number is too low, you will see errors like `"too many open files"` in the Portal logs. Note that this value might need to be even higher if other, non-Portal, pods are sharing the same user ID.

The Portal endpoints values are used when you configure a Portal service in the Cloud Manager. See [Registering a Portal service](#).

- `portalAdminEndpoint` - This is the Management Endpoint that is defined in Cloud Manager, which is used for communicating with API Manager.
- `portalUIEndpoint` - This is the Portal website URL defined in Cloud Manager. It determines the URL for the site that is created for each catalog. It is used for public access to the Portal from a browser.

Important: The endpoints defined here must be the ones used by the Management subsystem when registering the portal, and portal users when accessing portal sites. Do not attempt to proxy these endpoints. You will not be able to register the portal service nor access portal sites using any other endpoint URLs than those defined here.

Procedure

- Edit the `portal_cr` template CR to replace the placeholders with values for your deployment.

`$APP_PRODUCT_VERSION`

API Connect application version for the subsystems.

```
version: <version_number>
```

Example version number: 10.0.5.4

`$PROFILE`

Specify your portal subsystem profile, where `n` indicates number of replicas, `c` number of cores, and `m` is the minimum memory allocation in GB. For more information on profiles, see [Planning your deployment topology](#).

`$SECRET_NAME`

Use for image pull.

```
imagePullSecrets:  
- apic-registry-secret
```

`$DOCKER_REGISTRY`

The host name of the Docker Registry to which you uploaded the installation images. For example:

```
my.docker.registry.domain.example.com.
```

`$INGRESS_CLASS`

The ingress class that you want the endpoint to use. This is an optional property and if not specified, the ingress class with annotation `ingressclass.kubernetes.io/is-default-class: true` is used. If such an ingress class does not exist in the Kubernetes environment, then `nginx` is used. If you do set this value, it must refer to a valid ingress class configured in your Kubernetes system.

Note: This property is commented out in the template CR file. If you set this value, make sure to also uncomment it.

`$STACK_HOST`

The desired ingress subdomain for the API Connect stack. Used when you specify endpoints. Domain names that are used for endpoints cannot contain the underscore `"_"` character. You can do one of the following:

- Subdomain customization only
 - Accept the prefixes predefined for the ingress hostnames to use and replace all instances of `STACK_HOST` to be the desired ingress subdomain for the API Connect stack. For example, for a subdomain of `myhost.subnet.example.com`:

```
portalAdminEndpoint:  
  < ... >  
  hosts:  
    - name: api.portal.myhost.subnet.example.com  
      secret: portal-admin  
  
portalUIEndpoint:  
  < ... >
```

```

hosts:
- name: portal.myhost.subnet.example.com
  secret: portal-web

```

- Complete hostname customization
Change both the predefined prefixes and the `STACK_HOST` subdomain to match your desired hostnames.

For example, for `portalAdminEndpoint`, you can replace `api.portal.STACK_HOST` with `my.api.portal.myhost.subnet.example.com`, where `my.api.portal` replaces `api.portal`, and `myhost.subnet.example.com` replaces `STACK_HOST`:

```

portalAdminEndpoint:
  < ... >
  hosts:
  - name: my.api.portal.myhost.subnet.example.com
    secret: cm-endpoint

```

You can do this for some or all of the hostnames, depending on your customization requirements.

`STORAGE_CLASS`

The Kubernetes storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: `kubectl get sc`. For example, for local storage:

```

databaseVolumeClaimTemplate:
  storageClassName: STORAGE_CLASS
  volumeSize: 120Gi
databaseLogsVolumeClaimTemplate:
  storageClassName: STORAGE_CLASS
  volumeSize: 12Gi
webVolumeClaimTemplate:
  storageClassName: STORAGE_CLASS
  volumeSize: 200Gi
backupVolumeClaimTemplate:
  storageClassName: STORAGE_CLASS
  volumeSize: 300Gi
adminVolumeClaimTemplate:
  storageClassName: STORAGE_CLASS
  volumeSize: 20Gi
certVolumeClaimTemplate:
  storageClassName: STORAGE_CLASS
  volumeSize: 4Gi

```

`PLATFORM_CA_SECRET`

Set to `ingress-ca`. The `PLATFORM_CA_SECRET` contains the CA certificate for the platform REST API endpoint. The portal makes calls to the platform REST API and this property must be set so that the portal can verify the server certificate.

```

mgmtPlatformEndpointCASecret:
  secretName: PLATFORM_CA_SECRET

```

`CONSUMER_CA_SECRET`

Set to `ingress-ca`. The `CONSUMER_CA_SECRET` contains the CA certificate for the consumer REST API endpoint. The portal makes calls to the consumer REST API and this property must be set so that the portal can verify the server certificate.

```

mgmtConsumerEndpointCASecret:
  secretName: CONSUMER_CA_SECRET

```

Note:

- The `databaseVolumeClaimTemplate` has a minimum value of 30Gi. The `webVolumeClaimTemplate` has a minimum value of 20Gi. The `backupVolumeClaimTemplate` has a minimum value of 30Gi. However, all three need to be sized according to the installation.
- The `adminVolumeClaimTemplate` and `databaseLogsVolumeClaimTemplate` in this example have their minimum values and you do not need to change them.
- The `certVolumeClaimTemplate` is used by the `nginx` pod and in `2dcha` (two data center high availability) mode it is also by the `tunnel`, `db-remote` and `www-remote` pods. It can be sized small, just like the `adminVolumeClaimTemplate`, and does not need to be increased based on the number of sites.

2. Edit the `license` setting:

- Set `accept`: to `true` to accept the license. Note that the default value is `false`. If you do not accept the license, the Operator will not install the subsystem.
- Set `metric`: to track your product usage. Enter the unit of measure that is used for your program license:
 - `PROCESSOR_VALUE_UNIT` - Default value. If you leave the field blank, this value is used.
 - `MONTHLY_API_CALL` - Applies only to the IBM API Connect Hybrid Entitlement program.
 For information on tracking monthly call volume, see [Tracking API volume for auditing and compliance](#).
- Set `use`: to either `production` or `nonproduction`, to match the license you purchased.
- Set `license`: to the License ID for the version of API Connect that you purchased. See [API Connect licenses](#).

Example entry to accept the license for a production system:

```

license:
  accept: true
  metric: PROCESSOR_VALUE_UNIT
  use: production
  license: L-RJON-BZ5LSE

```

3. Optional: If you want to disable mTLS for communications between the management and portal, and enable JWT instead, then add and set the properties `mtlsValidateClient` and `jwtksUrl`.

```

spec:
  ...
  mtlsValidateClient: false
  jwtksUrl: <JWTKS URL>

```

where `<JWTKS URL>` is the URL of the JWKS endpoint hosted on the management subsystems. To find out the `jwtksUrl`, describe the management CR and check the `status` section:

```
kubectl describe mgmt -n <namespace>
...
status:
- name: jwksUrl
  secretName: api-endpoint
  type: API
  uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information on JWT security, see [Enable JWT security instead of mTLS](#).

- Optional: If you want to use the [in-cluster](#) inter-subsystem communication feature, then add the `mgmtPlatformEndpointSvcCASecret` and the `mgmtConsumerEndpointSvcCASecret` to the `spec: section`; then uncomment the `mgmtPlatformEndpointSvcCASecret` and the `mgmtConsumerEndpointSvcCASecret` sections, and set the `secretName` for both:

```
mgmtPlatformEndpointSvcCASecret:
  secretName: management-ca # Usually management-ca
mgmtConsumerEndpointSvcCASecret:
  secretName: management-ca # Usually management-ca
```

Unless you customized your internal certificates, these secrets are always `management-ca`.

- Optional: If you are installing as part of a two data center disaster recovery set up, complete the steps in [Installing a two data center deployment on Kubernetes](#) before you apply the edited file.
- Install the Portal Custom Resource, replacing `<namespace>` with the target installation namespace in the Kubernetes cluster:

```
kubectl apply -f portal_cr.yaml -n <namespace>
```

- Verify that the Portal subsystem is fully installed:

```
kubectl get PortalCluster -n <namespace>
```

The installation is complete when `STATUS` reports `Running`, `Ready` reports that all services are online (3/3), and `MESSAGE` reports "Ready for API Cloud Manager service registration", for example:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	MESSAGE	AGE
portal	3/3	Running	<version>	<version>		Ready for API Cloud Manager service registration	2h

You do not need to wait for the portal installation to complete before you move on to the next subsystem installation.

- We highly recommend that you retain the Portal subsystem encryption secret in a safe location, in case you need to restore the Portal subsystem during a disaster recovery scenario.

The encryption secret is created when the Portal subsystem is initially deployed. You can obtain the name of the encryption secret from the Portal subsystem custom resource once the installation is complete. For example:

- Obtain the name of the encryptionSecret:

```
kubectl get ptl portal -o yaml | grep encryptionSecret
encryptionSecret: portal-enc-key
```

The name of encryption secret is `portal-enc-key`.

- Save the secret yaml in a safe location:

```
kubectl get secret portal-enc-key -o yaml > portal-enc-key.yaml
```

- Backup the Portal installation. See [Backing up and restoring the Developer Portal in a Kubernetes environment](#).

Note:

When you create or register a Developer Portal service, the Portal subsystem checks that the Portal web endpoint is accessible. However sometimes, for example due to the complexity of public and private networks, the endpoint cannot be reached. The following example shows the errors that you might see in the `portal-www` pod, admin container logs, if the endpoint cannot be reached:

```
An error occurred contacting the provided portal web endpoint: example.com
The provided Portal web endpoint example.com returned HTTP status code 504
```

In this instance, you can disable the Portal web endpoint check so that the Developer Portal service can be created successfully.

To disable the endpoint check, complete the following update:

On Kubernetes, OpenShift, and IBM® Cloud Pak for Integration

Add the following section to the Portal custom resource (CR) template:

```
spec:
  template:
    - containers:
      - env:
        - name: PORTAL_SKIP_WEB_ENDPOINT_VALIDATION
          value: "true"
        name: admin
      name: www
```

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you completed the installation of all the required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the [Cloud Manager configuration checklist](#).

- [Defining multiple portal endpoints for a Kubernetes environment](#)
Multiple public facing endpoints (`portal-www`) can be defined for the Developer Portal.

Defining multiple portal endpoints for a Kubernetes environment

Multiple public facing endpoints (portal-www) can be defined for the Developer Portal.

About this task

You can override the single endpoint definition for portal-WWW, and the associated portal-www-ingress TLS certificate, to support multiple portal-www endpoints.

For information about the endpoints for the Developer Portal, see [Installing the Developer Portal subsystem](#).

Following are the example endpoints for configuring different sites served by the same Developer Portal service, as configured in this task:

- <https://banking.example.com/loans>
- <https://insurance.example.com/vehicle>

These unique endpoints allow Developer Portal sites to be defined on the Developer Portal service with different host names and domains. They replace endpoints that distinguish different sites by sub paths, as shown in the following examples:

- <https://www.example.com/banking/loans>
- <https://www.example.com/insurance/vehicle>

Procedure

Edit your `portal_cr` template. You can add multiple entries for your endpoints in the `hosts` element:
For example:

```
spec:
  portalUIEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: pt1.host1.example.com
        secretName: portal-web-host1
      - name: pt1.host2.example.com
        secretName: portal-web-host2
      - name: pt1.host3.example.com
        secretName: portal-web-host3
```

Installing the Analytics subsystem

Install the IBM® API Connect analytics subsystem.

Review the section on [Planning your analytics deployment](#) to ensure that you understand the choices for configuring the topology, disk space, and other elements of the analytics subsystem.

Note: When the analytics service is configured to store data, it uses OpenSearch, which requires map counts higher than the operating system defaults. The minimum recommended value is **262144**. If you plan to store analytics data locally (that is, do you not intend to disable local storage and offload all data), increase the default map count on every Kubernetes node:

1. To change the map counts on the live system, run the following command on every Kubernetes node:

```
sudo sysctl -w vm.max_map_count=262144
```

2. To persist this change when node restarts occur, add the following setting to the `/etc/sysctl.conf` file:

```
vm.max_map_count = 262144
```

For more information, see [Important settings](#) in the OpenSearch documentation.

- [Creating the analytics CR](#)
Create the installation CR for IBM API Connect analytics so that you can deploy the analytics subsystem.
- [Topology settings and additional analytics features](#)
Update the Kubernetes analytics CR to configure additional analytics features.
- [Installing analytics](#)
Deploy the analytics CR to install the analytics subsystem in your IBM API Connect deployment.

Creating the analytics CR

Create the installation CR for IBM® API Connect analytics so that you can deploy the analytics subsystem.

About this task

To install the analytics subsystem, create a Custom Resource (CR) in a `.yaml` file. The file contains all of your configuration settings for the analytics deployment. Use the file to install, upgrade, and update the configuration of your analytics subsystem.

Note: The `analytics_cr.yaml` and `analytics_dedicated_cr.yaml` template files are stored in the directory where you extracted `helper_files.zip`.

Procedure

1. Create or copy the appropriate analytics CR file from the helper_files, depending on which deployment storage type you want, see: [Planning the analytics profile, storage class, and storage type](#).

- If you want to use shared storage, use analytics_cr.yaml.
- If you want to use dedicated storage use analytics_dedicated_cr.yaml. Dedicated storage is only supported on three replica deployments.

2. Edit the CR file and update the following settings:

You cannot install the analytics subsystem until the placeholders are replaced with real values for your environment.

\$APP_PRODUCT_VERSION

API Connect application version for the subsystems.

```
version: <version_number>
```

Example version number: 10.0.5.4

\$PROFILE

Specify your analytics subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see [Planning your deployment topology](#).

\$SECRET_NAME

The name of the secret used to pull images from the Docker registry.

```
imagePullSecrets:  
- apic-registry-secret
```

\$DOCKER_REGISTRY

The host name of the Docker Registry to which you uploaded the installation images. For example:

```
my.docker.registry.domain.example.com.
```

\$INGRESS_CLASS

The ingress class that you want the endpoint to use. This is an optional property and if not specified, the ingress class with annotation `ingressclass.kubernetes.io/is-default-class:`

`true` is used. If such an ingress class does not exist in the Kubernetes environment, then `nginx` is used. If you do set this value, it must refer to a valid ingress class configured in your Kubernetes system.

Note: This property is commented out in the template CR file. If you set this value, make sure to also uncomment it.

\$STACK_HOST

The desired ingress subdomain for the API Connect stack. Used when specifying the ingestion endpoint. Domain names that are used for endpoints cannot contain the underscore "_" character. You can customize the subdomain or the complete host name:

- Subdomain customization only
Accept the prefixes predefined for the ingress host names to use and replace all instances of `$STACK_HOST` with the desired ingress subdomain for the API Connect stack. For example, if your host is `myhost.subnet.example.com`:

```
ingestion:  
  endpoint:  
    < ... >  
    hosts:  
      - name: ai.myhost.subnet.example.com  
        secretName: analytics-ai-endpoint  
    < ... >
```

\$STORAGE_CLASS

The Kubernetes storage class to be used for persistent volume claims (for more information, see [Planning the analytics profile, storage class, and storage type](#)). Find the available storage classes in the target cluster by running the following command: `kubect1 get sc`.

- For shared storage deployments:

```
storage:  
  type: shared  
  shared:  
    volumeClaimTemplate:  
      storageClassName: local_storage
```

- For dedicated storage deployments, set `$STORAGE_CLASS` in two places:

```
storage:  
  type: dedicated  
  shared:  
    volumeClaimTemplate:  
      storageClassName: local_storage  
  ...  
  master:  
    volumeClaimTemplate:  
      storageClassName: local_storage  
  ...
```

\$DATA_VOLUME_SIZE

Replace `$DATA_VOLUME_SIZE` with the value that you calculated in [Data storage: calculate how much data you want to store](#). If not specified it defaults to 50Gi

```
storage:  
  ...  
  shared:  
    volumeClaimTemplate:  
      ...  
      volumeSize: 50Gi
```

3. Edit the `license` setting:

- Set `accept`: to `true` to accept the license. Note that the default value is `false`. If you do not accept the license, the Operator will not install the subsystem.
- Set `metric`: to track your product usage. Enter the unit of measure that is used for your program license:
 - `PROCESSOR_VALUE_UNIT` - Default value. If you leave the field blank, this value is used.
 - `MONTHLY_API_CALL` - Applies only to the IBM API Connect Hybrid Entitlement program.

For information on tracking monthly call volume, see [Tracking API volume for auditing and compliance](#).

c. Set `use`: to either `production` or `nonproduction`, to match the license you purchased.

d. Set `license`: to the License ID for the version of API Connect that you purchased. See [API Connect licenses](#).

Example entry to accept the license for a production system:

```
license:
  accept: true
  metric: PROCESSOR_VALUE_UNIT
  use: production
  license: L-RJON-CEBL97
```

4. Optional: If you want to disable mTLS for communications between the management and analytics subsystem, and between the gateway and analytics subsystem, and enable JWT instead, then add and set the properties `mtlsValidateClient` and `jwtUrl`.

```
spec:
  ...
  mtlsValidateClient: false
  jwtUrl: <JWKS URL>
```

where `<JWKS URL>` is the URL of the JWKS endpoint hosted on the management subsystems. To find out the `jwtUrl`, describe the management CR and check the `status` section:

```
kubectl describe mgmt -n <namespace>
...
status:
- name: jwtUrl
  secretName: api-endpoint
  type: API
  uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information on JWT security, see [Enable JWT security instead of mTLS](#).

Note: It is not possible to use JWT on the V5 compatible gateway to analytics message flow.

5. Save the file.

Results

You now have an analytics CR that is configured with the default analytics topology described in [Planning your topology](#).

What to do next

If you do not want to add any more configuration options, skip to the topic [Installing analytics](#).

Important: All additional analytics configuration options can be added, changed, or removed after installation **except** for disablement or enablement of internal storage. If you want to disable internal storage you must configure this in the analytics CR before installation: [Disabling internal storage](#)

Otherwise, add configuration options to the CR as explained in [Topology settings and additional analytics features](#)

Topology settings and additional analytics features

Update the Kubernetes analytics CR to configure additional analytics features.

About this task

Based on the decisions you made while [defining your topology](#), configure your deployment options by completing the steps in the following topics to enable the persistent queue, configure data offloading, use dedicated storage, and disable internal storage.

Important: All additional analytics configuration options can be added, changed, or removed after installation **except** for disablement or enablement of internal storage. If you want to disable internal storage you must configure this in the analytics CR before installation: [Disabling internal storage](#)

- [Analytics persistent queue](#)
For enhanced reliability in your analytics data pipeline, enable the persistent queue feature by updating your Kubernetes analytics CR.
- [Offloading data to a third-party system](#)
Update the Analytics CR to configure data offloading from your IBM API Connect Analytics deployment to a third-party system.
- [Disabling internal storage](#)
Update the Kubernetes analytics CR to disable local storage in your IBM API Connect analytics deployment.
- [Modifying incoming analytics data](#)
Update the analytics CR to customize data in your analytics deployment.
- [Reference: Fields in the Analytics CR](#)
Use the table of Analytics-specific fields to configure the CR for installing your IBM API Connect Analytics subsystem.

Analytics persistent queue

For enhanced reliability in your analytics data pipeline, enable the persistent queue feature by updating your Kubernetes analytics CR.

The Logstash persistent queue is an optional feature that provides enhanced reliability in the data pipeline. With the persistent queue enabled, the ingestion pod buffers incoming API event data to persistent storage so that no events are lost if the downstream API Connect services become unavailable.

For example, if the analytics storage service is restarted, any incoming API events are kept in persistent storage and sent on to the storage service when it is available again. If the persistent queue is not enabled, then the ingestion pod rejects further API event data coming from the gateway.

Another benefit of this feature is better handling of unexpected traffic spikes. The only disadvantages are higher CPU and persistent storage requirements.

To enable the persistent queue feature:

1. Edit your `analytics_cr.yaml` and add the following to the ingestion section.

```
2. ingestion:
  queue:
    type: persisted
    volumeClaimTemplate:
      storageClassName: <storage-class-name>
      volumeSize: <storage size>
```

Where:

- `<storage-class-name>` is the Kubernetes storage class defined for persistent volumes. For example `local-storage`
 - `<storage size>` is the maximum size for your persistent queue. The minimum value is `30Gi`.
3. Save the file. If updating these settings after installation, confirm that the analytics ingestion pod restarts.

Offloading data to a third-party system

Update the Analytics CR to configure data offloading from your IBM® API Connect Analytics deployment to a third-party system.

About this task

For information about offloading data from your Analytics deployment, see [Planning to offload data to a third-party system](#).

Procedure

1. Open the `analytics_cr.yaml` file for editing.
2. Add the following section to the `spec` section of the CR:

```
external:
  offload:
    enabled: true
    output: |
      $EXTERNAL_OFFLOAD_OUTPUT
```

where `$EXTERNAL_OFFLOAD_OUTPUT` is the plugin output for the target third-party system.

The following example enables offloading to HTTP:

```
external:
  offload:
    enabled: true
    output: |
      http {
        url => "example.com"
        http_method => "post"
        codec => "json"
        content_type => "application/json"
        id => "offload_http"
      }
```

For examples of output plugins, see [Sample output plugins for offloading data](#).

3. Optional: Modify the output plugin to include certificates for connecting to a secure endpoint. If your third-party endpoint requires a particular certificate, you can configure the certificate and then reference it in the offload plugin. Include the `secretName` as well as the `cacert`, `client_cert`, and `client_key` settings.

For example, the following output plugin enables offloading to HTTP, and provides information on the certificate that is needed for authenticating with the endpoint:

```
external:
  offload:
    secretName: offload-certificates
    enabled: true
    output: |
      http {
        url => "https://example.com:443"
        http_method => "post"
        codec => "json"
        content_type => "application/json"
        id => "offload_http"
        cacert => "/etc/velox/external_certs/offload/cacert.pem"
        client_cert => "/etc/velox/external_certs/offload/client.pem"
        client_key => "/etc/velox/external_certs/offload/client.key"
      }
```

The secret and its defined files are mounted to the `/etc/velox/external_certs/offload` directory in the ingestion pod container.

For information on creating a secret to store the certificates, see [Providing a custom certificate for Analytics offload](#).

4. Optional: Add a `filter` after the output plugin to modify data prior to offloading it. The following example filter removes the `query_string` field:

```

external:
  offload:
    enabled: true
    output: |
      http {
        url => "example.com"
        http_method => "post"
        codec => "json"
        content_type => "application/json"
        id => "offload_http"
      }
    filter: |
      ruby {
        remove_field => ["query_string"]
      }

```

For information about modifying offloaded data, see [Planning to modify your incoming analytics data](#).

5. Save the file. If updating these settings after installation, confirm that the analytics ingestion pod restarts.

- [Sample output plugins for offloading data](#)
Review sample output plugins to see how to code your own plugins for offloading data from IBM API Connect Analytics.
- [Providing a custom certificate for Analytics offload](#)
Prepare a private or self-signed certificate for offloading data and then add it to the IBM API Connect Analytics deployment on Kubernetes.

Sample output plugins for offloading data

Review sample output plugins to see how to code your own plugins for offloading data from IBM® API Connect Analytics.

Configure output plugins that direct analytics data to your third-party systems. An output plugin specifies where analytics data is offloaded, plus any information required to access that system, such as certificate information. The output plugin for displaying analytics data in the Analytics user interface is configured by default.

Note: If your third-party system requires a private or self-signed certificate, configure it as explained in [Providing a custom certificate for Analytics offload](#) and include a reference to the certificate in your output plugin. See the second HTTP output plugin for an example of how to reference the certificate.

HTTP

To configure data offload to an HTTP server, you must provide the server URL. You can optionally add standardized or custom HTTP headers to define additional operating parameters.

The following example configures an HTTP output plugin.

```

http {
  url => "example.com"
  http_method => "post"
  codec => "json"
  content_type => "application/json"
  headers => {
    "x-acme-index" => "8"
    "x-acme-key" => "0d5d259f-b8c5-4398-9e58-77b05be67037"
  }
  id => "offload_http"
}

```

The following example configures an HTTP output plugin that references a custom certificate for accessing the third-party endpoint.

```

http {
  url => "https://example.com:443"
  http_method => "post"
  codec => "json"
  content_type => "application/json"
  headers => {
    "x-acme-index" => "8"
    "x-acme-key" => "0d5d259f-b8c5-4398-9e58-77b05be67037"
  }
  id => "offload http"
  cacert => "/etc/velox/external_certs/offload/cacert.pem"
  client_cert => "/etc/velox/external_certs/offload/client.pem"
  client_key => "/etc/velox/external_certs/offload/client.key"
}

```

For more information on offloading to HTTP, refer to the Logstash documentation on configuring the [HTTP output plugin](#).

Elasticsearch

To configure data offload to an Elasticsearch cluster, you must provide one or more server URLs, define how indices should be created by specifying a dynamically configured name, specify the number of primary shards for storing the indexed documents, and specify a number of replica shards for redundancy. You can optionally specify server authentication credentials.

The naming syntax that you specify for your indices determines how many indices are created and how they are characterized. For example, you can define indices that are created based on a date pattern (for example, daily or weekly), or indices that are created to store data by provider organization or by API name.

The following example configures an Elasticsearch output plugin.

```

elasticsearch {
  hosts => "https://example.com:443"
}

```

```
index => "apiconnect"
ssl => true
cacert => "/etc/pki/tls/cert.pem"
}
```

For more information on offloading to Elasticsearch, refer to the Elasticsearch documentation on the [Elasticsearch output plugin](#).

Apache Kafka

To configure data offload to a Kafka cluster, you must provide host and port connection details for one or more servers, and the name of the Kafka topic to which you want to publish the offloaded data. For logging and monitoring purposes within Kafka, you can optionally specify a string identifier by which API Connect can be uniquely identified.

The following example configures a Kafka output plugin.

```
kafka {
  topic_id => "test"
  bootstrap_servers => "example.com:9093"
  codec => "json"
  id => "kafka_offload"
  ssl_truststore_location => "/usr/share/logstash/jdk/lib/security/cacerts"
  ssl_truststore_password => "changeit"
  ssl_truststore_type => "JKS"
  security_protocol => SSL
}
```

For more information on offloading to Apache Kafka, refer to the Logstash documentation on the [Kafka output plugin](#).

Syslog

To configure data offload to Syslog, you must provide host and port connection details for the server.

Restriction: Syslog does not support the use of chained client certificates for TLS profiles.

The following example configures a Syslog output plugin.

```
syslog {
  host => "example.com"
  port => 601
  protocol => "ssl-tcp"
  id => "offload_syslog"
  appname => "apiconnect"
  msgid => "%{org_id}"
  facility => "log audit"
  severity => "informational"
  codec => "json"
  ssl_cacert => "/etc/pki/tls/cert.pem"
}
```

Attention: Due to a third-party issue with Logstash, most of the facility values are not set correctly on the offload server. As a temporary workaround, set the facility to "user-level" to ensure that the corresponding value is correctly sent to the offload server. When Logstash corrects this problem, you must update the offload configuration to change the value to a more appropriate setting.

For more information on offloading to Syslog, refer to the Logstash documentation on the [Syslog output plugin](#).

OpenSearch

To configure data offload to OpenSearch: <https://opensearch.org/docs/latest/clients/logstash/ship-to-opensearch/>

Providing a custom certificate for Analytics offload

Prepare a private or self-signed certificate for offloading data and then add it to the IBM® API Connect Analytics deployment on Kubernetes.

Procedure

1. Log in to a server that has kubectl access to the Kubernetes cluster where the API Connect Analytics subsystem is deployed.
2. Encode the certificate file contents in base64.

You can use either JKS files or text-based files such as PEM files:

- If you use a JKS file for the certificate, you must encode it.
- If you use a text-based file such as PEM, then you can either encode it, or paste its contents directly.

There are various tools for base64 encoding. If the `cat` and `base64` commands are available you can run the following command to encode the file contents:

```
cat keystore.jks | base64
```

Copy the output so that you can paste it into the secret in the next step.

3. Create a secret to contain the certificate.
 - a. Create a file called `analytics_certs.yaml` to contain the secret.

The secret's name that you specify in the file does not have to match the file name. In the examples, the secret's name is `offload-certificates`. Do not use the underscore character (`_`) in the secret name.

The method for adding the certificate to the secret depends on whether you encoded the file contents in the previous step:

- Base64-encoded file: In the **data** section of the secret, type a name for the field (you can choose the name). Paste the encoded output from the previous step (enclosed in `" "`) as the value.

```
apiVersion: v1
kind: Secret
metadata:
  # Change value of name to be whatever you wish secret to be called
  name: offload-certificates
  # Only base64 encoded data should be placed in data section; JKS keys would go here
data:
  keystore.jks: "output_of_base64_encoded_jks"
```

- Plain text file: In the **stringData** section of the secret, type a name for the key or certificate (you can choose the name), and paste the value.

```
apiVersion: v1
kind: Secret
metadata:
  # Change value of name to be whatever you wish secret to be called
  name: offload-certificates
stringData:
  cacert.pem: |-
    -----BEGIN CERTIFICATE-----
    MIIDIDCCAomgAwIBAgIEND70zzANBgkqhkiG9w0BAQUFADBOMQswCQYDVQQGEwJVUzEQMA4GA1UE
    ChMHRXFlaWZheDEtMCsGA1UECXMkRXFlaWZheCBTZWN1cmUgQ2VydG1maWNhdGUgQXV0aG9yaXR5
    MB4XDTK4MDgyMjE2NDE1MVoXDTE4MDgyMjE2NDE1MVowTjE1MAkGA1UEBHMVVMxZDA0BGNVBAoT
    B0VxdWlmYXgxtLTAzBgNVBAStJEVxdWlmYXgguU2VjdXJlIEN1cnRpZmljYXR1IEF1dGhvcml0eTCB
    nzANBgkqhkiG9w0BAQEFAAOBjQAwYkCgYEAwV2xWGCiYu6gmi0fCG2RFGiYCh7+2gRvE4RiIcPR
    fM6fBeC4AfBONOziipUEZKzxa1NfBbPLZ4C/QgKO/t0BCezhABRP/PvwDN1Dulsr4R+AcJkVV5MW
    8Q+XarfCaCMczE1ZMKxRHjuvK9buY0V7xd1fUNLjUA86iOe/FP3gx7kCAwEAaOCaQkwggEFMHAG
    A1UdHwRPMGcwZaBjoGGkXzBdmQswCQYDVQQGEwJVUzEQMA4GA1UEChMHRXFlaWZheDEtMCsGA1UE
    CxMkRkXFlaWZheCBTZWN1cmUgQ2VydG1maWNhdGUgQXV0aG9yaXR5MQ0wCwYDVQQDEwRDUkwMB0G
    A1UdEAQTMGBGDzIwMTgwODIyMTY0MTUxWjALBgNVHQ8EBAMCAQYwHwYDVR0jBBGwFoAUSOZo+SvS
    spXXR9gJ1BBPM5iQn9QwHQYDVR0OBBYEFjmaPkr0rKV10fYIyAQtzOYkJ/UMAwGA1UdEwQFMAMB
    Af8wGgYJKoZIhVZ9B0EABA0wCxsFVjmuMGMDAgbAMA0GCSqGSIb3DQEBAQUAA4GBAFjOKer89961
    zgK5FTWF0bnj4JXMJTENAKaSbn+2kmOeUJXRmm/kEd5jhW6Y7qj/WsjTVbJmcVfewCHRPSqnI0kB
    BIZCe/zuf6IWURVnZ9NA2zsmwLIodz2uFHdh1voqZiegDfqnclzqcPGUIWVEX/z87yloqaKHee95
    70+sB3c4
    -----END CERTIFICATE-----
```

You can add multiple keys/certificates, setting the field names for each as desired. For each key, a file based on the corresponding field name is created in the ingestion pod container at `/etc/velox/external_certs/offload`. The examples here would result in the files `keystore.jks` and `cacert.pem`.

b. Save the file.

4. Optional: Create an additional secret.

If you have sensitive data such as a certificate password that must be available in text format, you can store it in another secret.

If your password is not sensitive (such as the JKS default password), then you can supply it directly in the Logstash configuration, and skip this step.

- Create the file `offload_env_var.yaml`. The secret's name that you specify in the file does not have to match the file name. In this example, the secret's name is `offload-jks-password`.
- In the **stringData** section, set the property `jks_password.txt` to your password (in plain text):

```
apiVersion: v1
kind: Secret
metadata:
  name: offload-jks-password
stringData:
  jks_password.txt: <password>
```

- Run the following command to update the cluster:

```
kubectl apply -f offload_env_var.yaml -n your_namespace
```

Where `offload_env_var.yaml` is the secret's file name.

5. Apply the secret to the cluster where the Analytics subsystem is installed.

Run the following command to update the cluster:

```
kubectl apply -f file_name_of_secret.yaml -n namespace
```

Where:

- `file_name_of_secret.yaml` is the secret's file name.
- `namespace` is your analytics namespace.

6. Edit the Analytics CR and add a reference to the secret and passwordSecret in the **offload** section:

```
offload:
  enabled: true
  output: |
    kafka {
      ...
    }
  secretName: offload-certificates
  passwordSecretName: offload-jks-password
```

7. If you want to reference the certificate in an **offload plugin**, configure the plug-in now.

Step 3 of that procedure shows how you can include the certificate information in the plug-in.

Disabling internal storage

Update the Kubernetes analytics CR to disable local storage in your IBM® API Connect analytics deployment.

About this task

The Analytics subsystem provides a complete solution for routing, storing, and viewing the analytics data. However, you might not need the internal storage and viewing solution, especially if you are primarily using a third-party system for data offloading. In this scenario, you can greatly reduce your CPU and memory costs by disabling the internal storage and viewing components.

With this approach, data that is offloaded is not viewable in the API Manager and Developer Portal UIs.

When you disable internal storage for analytics data, the following microservices are disabled:

- `osinit`
- `storage`

Important: The disablement of internal storage must be set in the analytics CR yaml file before installation. It is not possible to disable or re-enable internal storage after installation.

Procedure

1. Required: Configure data offloading as explained in [Offloading data to a third-party system](#).
If you do not have offloading configured and also disable internal storage, you will have no analytics data.
2. Open the `analytics_cr.yaml` file for editing.
3. Locate the `storage` section and set `enabled` to `false`:

```
storage:
  enabled: false
```

4. Save the file.

Modifying incoming analytics data

Update the analytics CR to customize data in your analytics deployment.

Before you begin

Review the use of filters for modifying analytics data, and make sure that you understand the restrictions. For information about customizing data, see [Planning to modify your incoming analytics data](#).

About this task

You can define filters that add, modify, or remove fields from the incoming API event records, before they are stored in API Connect or offloaded to a third-party system.

Procedure

1. Open the `analytics_cr.yaml` file for editing.
2. Filters for API events that are offloaded to an external system are defined in the `external.offload` section. Filters for API events that are stored in the API Connect OpenSearch database are defined in the `ingestion` section.

Ingestion filters - for API event data stored by API Connect
Add the filter definition to the `ingestion` section:

```
spec:
  ingestion:
    ...
    filter: |
      mutate {
        remove_field => ["query_string"]
      }
```

Offload filters - for API event data that is sent to a third-party system.
Add the filter definition to the `external.offload` section:

```
spec:
  external:
    offload:
      filter: |
        mutate {
          remove_field => ["query_string"]
        }
```

For more examples, see [Sample filters for modifying analytics data](#).

3. Optional: Include geoIP fields in your analytics data, for both stored and offloaded data, by inserting `geoIPEnabled: true` in the ingestion section:

```
ingestion:
  ...
  geoIPEnabled: true
```

By default the `client_geoip` and `gateway_geoip` fields are not included in the analytics data. You do not need to add `geoIPEnabled` to the `external.offload` section. Enabling geoIP in the `ingestion` section enables it for both data routes.

Note: To ensure the value in the `client_geoip` field is accurate, the gateway must receive the `X-Forwarded-For` header in all API calls. Check with the administrator of your deployment environment to ensure that the `X-Forwarded-For` header is passed to your gateways. For example, in a Kubernetes environment where NINGX ingresses are used, configure the NGINX ingress that is used by your gateway to use the `X-Forwarded-For` header: [NGINX Configuration "use-forwarded-headers"](#).

4. Save the file. Changes to filters take effect after approximately 30 seconds.

- **Sample filters for modifying analytics data**

Review sample filters to see how to code your own filters for refining IBM API Connect Analytics data by adding fields, removing fields, or modifying field contents.

Sample filters for modifying analytics data

Review sample filters to see how to code your own filters for refining IBM® API Connect Analytics data by adding fields, removing fields, or modifying field contents.

You can use these sample filters in the `ingestion.filter` and the `external.offload.filter` configurations.

Adding a new field

You can add custom data to API event data by adding a field to the logging document. For example, if you want to search or visualize data that is stored in request and response headers, you can add fields to a filter to include that information.

To avoid naming conflicts with future or current analytics fields, include a prefix that ensures your new field name is unique. For example, instead of naming the field `employee_num` you might name it `x_mycompany_employee_num`.

The following example copies the contents of the `X-Employee-Num` field from the request header and adds it to the `x_mycompany_employee_num` field in the event data. Adding the field to the event data enables you to use the information in visualizations.

```
if [request_http_headers] {
  ruby {
    code => "event.get(['request_http_headers']).collect {|i| event.set(['x_mycompany_employee_num'], i['X-Employee-Num']) if i.has_key?('X-Employee-Num')}"
  }
}
```

Modifying an existing field

Sometimes you don't want to remove a field entirely from your data, you just want to redact sensitive information such as IDs to prevent them from being exposed in visualizations. You can modify the contents of a field and replace information with a symbols or a message.

Remember: The following fields should not be modified if the data is being written to internal analytics storage: `org_id`, `catalog_id`, `space_id`, `developer_org_id`, `datetime`, and `@timestamp`.

The following example replaces sensitive information in the `X-Employee-Name` and `X-Employee-ID` request headers with the string: `*****sanitized*****`.

```
if [request_http_headers] {
  ruby {
    code => "headers=['X-Employee-Name','X-Employee-ID']; newHeaders = event.get(['request_http_headers']).collect {|i| headers.each {|header| i[header] = '*****sanitized*****' if i.has_key?(header)}; i}; event.set(['request_http_headers'], newHeaders)"
  }
}
```

Removing an existing field

Use the mutate `remove_field` operation to delete a field from the Analytics data. To remove multiple fields, delimit the field names with commas.

The following example removes multiple fields (`request_http_headers`, `response_http_headers`, `request_body`, `response_body`, and `query_string`) from the Analytics data:

```
mutate {
  remove_field => ["request_http_headers", "response_http_headers", "request_body", "response_body", "query_string"]
}
```

Combining filters with offload and ingestion configuration

Offload events from `catalog1` only:

```
offload:
  enabled: true
  if [catalog_name] == "offload1" {
    http {
      url => "http://offloadhost.example.com/offload"
      http_method => "post"
      codec => "json"
      content_type => "application/json"
      id => "offload_http"
    }
  }
}
```

Offload to different locations based on catalog name:

```

offload:
  enabled: true
  output: |
    if [catalog_name] == "offload1" {
      http {
        url => "http://offloadhost.example.com /offload1"
        http_method => "post"
        codec => "json"
        content_type => "application/json"
        id => "offload1_http"
      }
    }
    else if [catalog_name] == "offload2" {
      http {
        url => "http://offloadhost2.example.com /offload2"
        http_method => "post"
        codec => "json"
        content_type => "application/json"
        id => "offload2_http"
      }
    }
  }
}

```

Use the catalog name as a variable in the offload location:

```

offload:
  enabled: true
  http {
    url => "http://offloadhost.example.com /%{[catalog_name]}"
    http_method => "post"
    codec => "json"
    content_type => "application/json"
    id => "offload_http"
  }
}

```

Do not offload events from `apic` and `sandbox` catalogs:

```

offload:
  enabled: true
  if [catalog_name] !~ /sandbox|apic/ {
    http {
      url => "http://offloadhost.example.com /%{[catalog_name]}"
      http_method => "post"
      codec => "json"
      content_type => "application/json"
      id => "offload_http"
    }
  }
}

```

Do not store events from `apic` and `sandbox` catalogs:

```

ingestion:
  filter: |
    if [catalog_name] =~ /sandbox|apic/ {
      drop { }
    }
}

```

Filter and output plugin for splunk

When you create a filter for use with splunk, you must also create an output plugin for it.

Filter:

```

if "apicapievent" in [tags] {
  ruby {
    code => "event.set('@metadata[newevent]', event.to_json)"
  }
}

```

Output plugin:

```

http {
  url => "http://your-domain/services/collector/event"
  http_method => "post"
  codec => "json"
  content_type => "application/json"
  id => "offload_http"
  format => "message"
  message => '{"event": "%{[@metadata][newevent]}", "index": "applications-int-ms", "sourcetype": "json:apimlogs"}'
  headers => ["Authorization", "Splunk XXXX"]
}

```

Reference: Fields in the Analytics CR

Use the table of Analytics-specific fields to configure the CR for installing your IBM® API Connect Analytics subsystem.

The two fields that must be in Analytics CR are `ingestion.endpoint` and `ingestion.clientSubjectDN`. The other fields are optional depending on the analytics features you want to configure.

Table 1. Configurable settings in the Analytics subsystem CR.

Field	Type	Default value	Description	Comments
allowUpgrade	Boolean	no	Set to true to allow analytics upgrade to v10.0.5.	See Additional considerations upgrading to analytics v10.0.5 - Kubernetes , Additional considerations upgrading to analytics v10.0.5 - OpenShift .
databaseBackup: host	String		The URL to your S3 storage.	See Configuring backup settings for Analytics , Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration .
databaseBackup: path	String		The path to your S3 storage at the URL specified by <code>databaseBackup.host</code> .	See Configuring backup settings for Analytics , Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration .
databaseBackup: schedule	String		Specifies the frequency of automatic backups. Use any valid <code>cron</code> string.	See Configuring backup settings for Analytics , Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration .
databaseBackup: enableCompression	Boolean	true	Specifies whether to compress the backups.	See Configuring backup settings for Analytics , Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration .
databaseBackup: credentials	String		The name of your S3 storage authentication secret.	See Configuring backup settings for Analytics , Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration .
databaseBackup: chunkSize	String	1Gi		See Configuring backup settings for Analytics , Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration .
databaseBackup: enableServerSideEncryption	Boolean			See Configuring backup settings for Analytics , Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration .
external: offload: filter	String		Filter configuration for external offload.	See Offloading data to a third-party system - Kubernetes , Offloading data to a third-party system - OpenShift .
external: offload: enabled	Boolean	false	Enables analytics offload feature.	See Offloading data to a third-party system - Kubernetes , Offloading data to a third-party system - OpenShift .
external: offload: output	String		Output configuration for external offload.	See Offloading data to a third-party system - Kubernetes , Offloading data to a third-party system - OpenShift .
external: offload: passwordSecretName	String		The name of the secret that has the password to access the certificate keystore.	For where you are using a password protected key/truststore. See Offloading data to a third-party system - Kubernetes , Offloading data to a third-party system - OpenShift .
external: offload: secretName	String		The name of the secret that contains the certificates used for connecting to the offload target.	See Offloading data to a third-party system - Kubernetes , Offloading data to a third-party system - OpenShift .
ingestion: endpoint	Object		FQDN and TLS Secrets for client endpoint.	Identifies the endpoint that enables the Gateway to push your analytics data.
ingestion: clientSubjectDN	String		Subject DN (distinguished name) for authenticating with ingestion endpoint (RFC 2253).	
ingestion: filter	String		Filter to modify API event data that is stored internally (not offloaded).	See Configuring ingestion filters on Kubernetes , Configuring ingestion filters on OpenShift .
ingestion: queue	Object		Persistent queue settings. Must be set to persisted if present.	See Configuring persistent queue on Kubernetes , Configuring persistent queue on OpenShift .
ingestion: queue: type	String		Defines the storage type for the persistent storage feature. Options are 'persisted' or 'memory'	See Configuring persistent queue on Kubernetes , Configuring persistent queue on OpenShift .
ingestion: queue: volumeClaimTemplate: volumeSize	String		Storage size for your persistent queue.	See Configuring persistent queue on Kubernetes , Configuring persistent queue on OpenShift .

Field	Type	Default value	Description	Comments
<code>ingestion: queue:</code> <code>volumeClaimTemplate:</code> <code>storageClassName</code>	String		Storage class name for your persistent storage.	See Configuring persistent queue on Kubernetes , Configuring persistent queue on OpenShift .
<code>ingestion: pipelineWorkers</code>	4 <= integer <= 128	24	Number of pipeline workers.	More workers mean more CPU and memory requirements, but gives you the benefit of extra threads to pull data off the ingestion http interface and process it for storage. This has diminishing returns as the size increases, and should usually be multiple of the number of cores on your hardware. Adjust with caution.
<code>ingestion: pipelineBatchSize</code>	10 <= integer <= 10000	250	Batch size of a pipeline worker.	Pulling more or less data off the http interface per worker can have benefits. Recommended values are 125, 250, and 500. Increasing the value requires more memory.
<code>ingestion: pipelineBatchDelay</code>	10 <= integer <= 5000	50	Batch delay of a pipeline worker.	How long to wait for the batch size to fill before leaving the http interface to process for storage. Having threads wait too long causes idle threads in your system; however, setting this value too low decreases performance. Adjust with caution.
<code>ingestion: configReloadAutomatic</code>	Boolean	true	Whether to automatically reload the pipeline on configuration changes.	Typically, when the underlying files for the pipeline are modified, the pipeline process automatically reloads and picks up the changes. However, in Kubernetes environments where the pipeline is mapped to configmaps, this is unreliable. Ingestion microservice restarts are usually required.
<code>ingestion: pipelineUnsafeShutdown</code>	Boolean	false	Whether to allow unsafe shutdown of the pipeline.	Typically, the pipeline attempts to empty all workers' batches before shutdown to prevent data loss. You might need to enable this setting if the pipeline is badly configured and refuses to respond to shut down requests (uncommon). Adjust with caution.
<code>standaloneCluster</code>	Boolean			
<code>storage: type</code>	String	shared	Type of storage deployment.	Shared is the only available option.
<code>storage: shared: volumeClaimTemplate:</code> <code>storageClassName</code>	String	cluster default	Storage class name.	Name of the storage class configured to use for creating the microservice persistent volume.
<code>storage: shared: volumeClaimTemplate: volumeSize</code>	String	50Gi	Volume size.	Size of the volume used for data storage. See Estimating internal storage space for more information.

Installing analytics

Deploy the analytics CR to install the analytics subsystem in your IBM® API Connect deployment.

Before you begin

Now that you have a complete analytics custom resource file, you should double-check it for accuracy. In particular, verify that:

- No placeholder values remain in the `analytics_cr.yaml` file.
- You configured all the topology options that you plan to deploy.
- You configured appropriate volume size for storage.

Procedure

1. Run the following command to apply the `analytics_cr.yaml`:

```
kubectl apply -f path/to/analytics-cr -n namespace
```

where:

- `path/to/analytics-cr` is the path to your saved `analytics_cr.yaml` file
- `namespace` is the name of your namespace

2. Verify that the analytics subsystem is fully installed by running the following command:

```
kubectl get AnalyticsCluster -n namespace
```

where `namespace` is the name of your namespace.

The installation is complete when the **READY** status is **True** and the **SUMMARY** reports that all services are online. Example:

```

NAME      READY  SUMMARY  VERSION  RECONCILED  VERSION  AGE
analytics True    5/5      <version> <version-build>

```

The **SUMMARY** total count is based on your configured topology. Both numbers are be equal when the installation is complete.

Results

Table 1 lists the number of microservices that should deploy, along with notes on which topologies might not include each microservice.

Table 1. Number of microservices deployed in typical Analytics subsystems

Expected	Microservice	Note
3	director	Always expected
3	ingestion	Always expected
3	mtls-gw	Always expected
3	storage	Not expected if you disabled internal storage
3	storage-os-master	Not expected if you disabled internal storage. Not expected if using shared storage.
3	osinit	Expected to be in Completed state

The one replica profile only expects 1 (instead of 3) of each microservice.

Installing a two data center deployment on Kubernetes

Additional installation instructions for a two data center disaster recovery (2DCDR) deployment on Kubernetes.

Before you begin

Ensure that you understand the concepts of 2DCDR in API Connect. For more information, see [Two data center deployment strategy on Kubernetes and OpenShift](#).

Download the operator files and custom resource definitions for use during installation onto both of your data centers. Ensure that you complete the instructions for [Deploying operators and cert-manager](#), and set the same ingress-ca certificate in both data centers: [Installing cert-manager and certificates in a two data center deployment](#).

Familiarize yourself with the instructions in [Installing the API Connect subsystems](#). Follow these instructions along with the 2DCDR specific steps in this topic.

Important: Each data center must use a different backup path for your Management backups. For portal backups each data center must use the same backup path. For more information, see [Backup and restore requirements for a two data center deployment](#).

Restrictions:

- API Connect is not supported on a FIPS-enabled environment.
- It is not possible to use the Automated API behavior testing application ([Installing the Automated API behavior testing application](#)) in a 2DCDR configuration ([Two data center deployment strategy on Kubernetes and OpenShift](#)).

The following endpoints must be the same on both data centers:

Management subsystem endpoints

- `cloudManagerEndpoint`
- `apiManagerEndpoint`
- `platformAPIEndpoint`
- `consumerAPIEndpoint`

Portal subsystem endpoints

- `portalAdminEndpoint`
- `portalUIEndpoint`

For more information about how to set the endpoints, see [Installing the Management subsystem cluster](#) on Kubernetes, [Installing the Developer Portal subsystem](#) on Kubernetes.

About this task

A 2DCDR deployment differs from a standard API Connect deployment in that the custom resources (CRs) for both the management and portal in each data center include a `multiSiteHA` configuration section. The `multiSiteHA` section is used to define which data center is the active and which is the warm-standby.

The following examples show `multiSiteHA` sections in the `PortalCluster` CRs for the active and warm-standby data centers.

<pre>siteName: dallas multiSiteHA: mode: active replicationEndpoint: annotations: cert-manager.io/issuer: ingress-issuer hosts: - name: ptlreplicationdallas.cluster1.example.com secretName: dallas-ptl-replication-worker-1 replicationPeerFQDN: ptlreplicationraleigh.cluster2.example.com tlsClient: secretName: ptl-replication-client ...</pre>	<pre>siteName: raleigh multiSiteHA: mode: passive replicationEndpoint: annotations: cert-manager.io/issuer: ingress-issuer hosts: - name: ptlreplicationraleigh.cluster2.example.com secretName: raleigh-ptl-replication-worker-1 replicationPeerFQDN: ptlreplicationdallas.cluster1.example.com tlsClient: secretName: ptl-replication-client ...</pre>
---	--

Table 1. `multiSiteHA` configuration properties

Configuration property	Description
------------------------	-------------

Configuration property	Description
<code>mode</code>	The state of the data center in the context of the 2DCDR deployment. Valid values are: <ul style="list-style-type: none"> <code>active</code> - indicates the active data center. <code>passive</code> - indicates the warm-standby data center.
<code>replicationEndpoint</code> <code>annotations:</code> <code>cert-manager.io/issuer:</code> <code>ingress-issuer</code> <code>hosts:</code> <code>- name:</code> <code>secretName:</code>	Contains details of the external ingress name for the subsystem in the current data center in the 2DCDR deployment. The hosts <code>name</code> is a unique fully qualified hostname that the other data center uses to communicate with the current data center.
<code>replicationPeerFQDN</code>	The ingress hostname for the other data center in the 2DCDR deployment. This information is required so that the two data centers can communicate with each other.
<code>tlsClient:</code> <code>secretName</code>	The TLS client secret for the subsystem.

The `siteName` property is a unique descriptive name for the portal and management subsystems, in each data center. For example, `dallas` in data center 1 (DC1), and `raleigh` in data center 2 (DC2).

- Set a different `siteName` on the active to the one set on the warm-standby data center.
- `siteName` is used in the hostnames, and so can contain only `a-z` and `0-9` characters.
- You can set a `siteName` in the `ManagementCluster` or `PortalCluster` CRs only on installation. You cannot change the name after deployment. If you don't set a `siteName` at first deployment, an automated `siteName` is created for you.

To install a 2DCDR deployment:

- Set the placeholder properties in the CR file for management and portal subsystems as detailed in [Installing the API Connect subsystems](#), but do not run the final `kubectl apply` operation. Complete the additional 2DCDR configuration steps before you run `kubectl apply`.
- Set the properties in the `multiSiteHA` section, and synchronize encryption secrets between the data centers as described in the procedure below:

Procedure

• Installing to an active data center

The following example shows how to set the multi-site HA CR values for deploying to the active data center. In this example, the initial active data center is called `dallas` (DC1) and the warm-standby data center is called `raleigh` (DC2).

- Set Dallas to be active for the API Manager service on DC1.
Edit the `ManagementCluster` CR file `management_cr` for DC1, and set the `multiSiteHA` properties as shown:

```
siteName: dallas
multiSiteHA:
  mode: active
  replicationEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: mgrreplicationdallas.cluster1.example.com
        secretName: dcl-mgmt-replication
  replicationPeerFQDN: mgrreplicationraleigh.cluster2.example.com
  tlsClient:
    secretName: dcl-mgmt-replication-client
```

- Set Dallas to be active for the Developer Portal service on DC1.
Edit the `PortalCluster` CR file `portal_cr` for DC1, and set the `multiSiteHA` properties as shown:

```
multiSiteHA:
  mode: active
  replicationEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: ptlreplicationdallas.cluster1.example.com
        secretName: dallas-ptl-replication-worker-1
  replicationPeerFQDN: ptlreplicationraleigh.cluster2.example.com
  tlsClient:
    secretName: ptl-replication-client
...
siteName: dallas
```

- Create a secret called `ptl-encryption-key` on the active and warm-standby data centers.
The secret must use the same random string in both data centers. Run the following command:

```
kubectl create secret generic ptl-encryption-key --from-literal=encryption_secret=<RANDOM STRING> -n <namespace>
```

The string can consist of uppercase letters, lowercase letters, and numbers, and must be at least 64 characters and no more than 100 characters.

- When the secrets are created on both DC1 and DC2, update the portal CR on DC1 and DC2 and add the following under the `spec` section:

```
encryptionSecret:
  secretName: ptl-encryption-key
```

Warning: Ensure that you add the secret directly under the `spec` section and not inside the `multiSiteHA` section.

- Create a secret called `mgmt-encryption-key` on the active and warm-standby data centers.
The secret must use the same random string in both data centers. Run the following command:
Note: Do not use the same random string that you used to create the secret for `ptl-encryption-key`.

```
kubectl create secret generic mgmt-encryption-key --from-literal=encryption_secret.bin=<RANDOM STRING> -n <namespace>
```

The string can consist of uppercase letters, lowercase letters, and numbers, and must be at least 64 characters and no more than 100 characters.

- When the secrets are created on both DC1 and DC2, update the portal CR on DC1 and DC2 and add the following under the `spec` section:

```
encryptionSecret:  
  secretName: mgmt-encryption-key
```

Warning: Ensure that you add the secret directly under the `spec` section and not inside the `multiSiteHA` section.

- Apply the CR files to the subsystems in DC1.

For example, to apply the `ManagementCluster` CR file to the DC1 cluster, run the following command:

```
kubectl apply -f management_cr.yaml -n <namespace>
```

Where `<namespace>` is the target installation namespace in the Kubernetes cluster for DC1. Repeat this process to apply the updated `PortalCluster` CR file to DC1.

You can verify that the installation process is running with the following command:

```
kubectl get ManagementCluster -n <namespace>
```

Change `ManagementCluster` to `PortalCluster` to verify the Developer Portal installation. For more information, see [Installing the API Connect subsystems](#).

- Set your dynamic router to direct all traffic to DC1. The endpoints that must be directed to DC1 are:

Management subsystem endpoints

- `cloudManagerEndpoint`
- `apiManagerEndpoint`
- `platformAPIEndpoint`
- `consumerAPIEndpoint`

Portal subsystem endpoints

- `portalAdminEndpoint`
- `portalUIEndpoint`

- **Installing the warm-standby data center**

The following example shows how to set the multi-site HA CR values for the remote data center called `raleigh` (DC2) to be the warm-standby. Use the same encryption key on both sites.

Ensure that the installation of the active data center is complete before you start the installation of the warm-standby data center.

- Set Raleigh to be warm-standby for the API Manager service on DC2.

Edit the `ManagementCluster` CR file `management_cr` for DC2, and set the `multiSiteHA` properties as shown:

```
multiSiteHA:  
  mode: passive  
  replicationEndpoint:  
    annotations:  
      cert-manager.io/issuer: ingress-issuer  
    hosts:  
      - name: mgrreplicationraleigh.cluster2.example.com  
        secretName: raleigh-mgr-replication-worker-1  
  replicationPeerFQDN: mgrreplicationdallas.cluster1.example.com  
  tlsClient:  
    secretName: mgr-replication-client  
  ...  
siteName: raleigh
```

- Set Raleigh to be warm-standby for the Developer Portal service on DC2.

Edit the `PortalCluster` CR file `portal_cr` for DC2, and set the `multiSiteHA` properties as shown:

```
multiSiteHA:  
  mode: passive  
  replicationEndpoint:  
    annotations:  
      cert-manager.io/issuer: ingress-issuer  
    hosts:  
      - name: ptlreplicationraleigh.cluster2.example.com  
        secretName: raleigh-ptl-replication-worker-1  
  replicationPeerFQDN: ptlreplicationdallas.cluster1.example.com  
  tlsClient:  
    secretName: ptl-replication-client  
  ...  
siteName: raleigh
```

- Apply the CR files to the subsystems in DC2.

For example, to apply the `ManagementCluster` CR file to the DC2 cluster, run the following command:

```
kubectl apply -f management_cr.yaml -n <namespace>
```

Where `<namespace>` is the target installation namespace in the Kubernetes cluster for DC2. Repeat this process to apply the updated `PortalCluster` CR file to DC2.

You can verify that the installation process is running with the following command:

```
kubectl get ManagementCluster -n <namespace>
```

Change `ManagementCluster` to `PortalCluster` to verify the Developer Portal installation. For more information, see [Installing the API Connect subsystems](#).

- **Validating your two data center deployment**

While the management subsystems on the warm-standby and active data centers are synchronizing their databases, the management status reports `Warning`, and the `haStatus` reports `pending`:

```
kubectl get mgmt -n <namespace>
```

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	MESSAGE
management	n/n	Warning	10.0.5.3-0	10.0.5.3-0		Management is ready. HA Status Warning - see HAStatus in CR for details

```
status:
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication not working, install or upgrade in progress.",
    "reason": "na",
    "status": "True",
    "type": "Pending"
  }
```

When the management database replication between sites is complete, the management status reports **Running**, and `status.haStatus` reports **Ready**:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	MESSAGE
management	n/n	Running	10.0.5.3-0	10.0.5.3-0		Management is ready.

```
status:
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication is working",
    "reason": "na",
    "status": "True",
    "type": "Ready"
  }
```

If the management database replication between sites fails for any reason other than because an install or upgrade is in progress, the `haStatus` output shows:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	MESSAGE
management	n/n	Warning	10.0.5.3-0	10.0.5.3-0		Management is ready. HA Status Warning - see HAStatus in CR for details

```
status:
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication not working",
    "reason": "na",
    "status": "True",
    "type": "Warning"
  }
```

If the warning persists, see [Troubleshooting a two data center deployment](#).

You can validate that your portal deployments are synchronizing by running `kubectl get pods` on both the active and warm-standby data centers, and confirming that the number and names of the pods all match (the UUIDs in the names might be different on each site), and that all are in the **Ready** state.

For additional replication verification checks, see [Verifying replication between data centers](#). It is recommended to run a test failover, and confirm that all of the expected data is present and correct on the newly active site. See [How to failover API Connect from the active to the warm-standby data center](#) for details.

- [Converting a single data center to a two data center deployment on Kubernetes](#)
Instructions on how to convert a single data center to a two data center disaster recovery (2DCDR) deployment on Kubernetes.

Related concepts

- [Maintaining a two data center deployment](#)
- [Backup and restore requirements for a two data center deployment](#)
- [Recovering from a failover of a two data center deployment](#)
- [Removing a two data center deployment](#)

Related tasks

- [How to failover API Connect from the active to the warm-standby data center](#)

Converting a single data center to a two data center deployment on Kubernetes

Instructions on how to convert a single data center to a two data center disaster recovery (2DCDR) deployment on Kubernetes.

Before you begin

Ensure that you understand the concepts of 2DCDR in API Connect. For more information, see [Two data center deployment strategy on Kubernetes and OpenShift](#).

Download the operator files and custom resource definitions for use during installation onto both of your data centers. Ensure that you complete the instructions for [Deploying operators and cert-manager](#), and set the same ingress-ca certificate in both data centers: [Installing cert-manager and certificates in a two data center deployment](#).

Familiarize yourself with the instructions in [Installing the API Connect subsystems](#). Follow these instructions along with the 2DCDR specific steps in this topic.

Important: Each data center must use a different backup path for your Management backups. For portal backups each data center must use the same backup path. For more information, see [Backup and restore requirements for a two data center deployment](#).

Restrictions:

- API Connect is not supported on a FIPS-enabled environment.
- It is not possible to use the Automated API behavior testing application ([Installing the Automated API behavior testing application](#)) in a 2DCDR configuration ([Two data center deployment strategy on Kubernetes and OpenShift](#)).

About this task

You can convert a single data center API Connect to a two data center disaster recovery deployment. Both data centers must have the same `portal-admin` and `portal-www` endpoints. The main difference is that the custom resource (CR) files for both the Management subsystem and the Developer Portal subsystem on each data center must include a `multiSiteHA` configuration section, and it's this section that determines the two data center DR deployment. It's also this section that's used to control the deployment in the future, for example in failover and recovery situations.

The following codeblocks show examples of completed `multiSiteHA` sections in the `PortalCluster` CR files for two data centers, one active and one warm-standby. Both the `ManagementCluster` and the `PortalCluster` CR templates have a `multiSiteHA` section containing the placeholder `$MULTI_SITE_HA`, and it is this placeholder that must be replaced with the values for your deployment.

<pre>siteName: dallas multiSiteHA: mode: active replicationEndpoint: annotations: cert-manager.io/issuer: ingress-issuer hosts: - name: ptlreplicationdallas.cluster1.example.com secretName: dallas-ptl-replication-worker-1 replicationPeerFQDN: ptlreplicationraleigh.cluster2.example.com tlsClient: secretName: ptl-replication-client ... </pre>	<pre>siteName: raleigh multiSiteHA: mode: passive replicationEndpoint: annotations: cert-manager.io/issuer: ingress-issuer hosts: - name: ptlreplicationraleigh.cluster2.example.com secretName: raleigh-ptl-replication-worker-1 replicationPeerFQDN: ptlreplicationdallas.cluster1.example.com tlsClient: secretName: ptl-replication-client ... </pre>
--	---

The configuration properties are described in the following table.

Table 1. The configuration properties of the multi-site HA section in the CR file

Configuration property	Description
<code>mode</code>	The state of the data center in the context of the two data center deployment. Valid values are: <ul style="list-style-type: none"> • <code>active</code> - indicates the primary data center • <code>passive</code> - indicates the warm-standby data center
<code>replicationEndpoint</code> <code>annotations:</code> <code>cert-manager.io/issuer</code> <code>hosts:</code> <code>- name:</code> <code>secretName:</code>	Contains details of the external ingress name for the subsystem in the current data center in the two data center deployment. The hosts <code>name</code> is a unique fully qualified hostname that the other data center uses to communicate with the current data center.
<code>replicationPeerFQDN</code>	The ingress hostname for the other data center in the two data center deployment. This information is required so that the two data centers can communicate with each other.
<code>tlsClient:</code> <code>secretName</code>	The TLS client secret for the subsystem.
<code>siteName</code>	A descriptive name for the Portal subsystem, and the Management subsystem, in the current data center, for example <code>dallas</code> . This name is used in the hostnames, and so can contain only <code>a-z</code> and <code>0-9</code> characters. Note that you can set a <code>siteName</code> in the <code>ManagementCluster</code> or <code>PortalCluster</code> CR only when the subsystems are first deployed, and you cannot change the name after deployment. If you don't set a <code>siteName</code> at first deployment, an automated <code>siteName</code> is created for you. For example, if you are converting an existing single Management cluster to a two data center Management cluster by adding the multi-site HA values, you will not be able to configure a <code>siteName</code> for the existing Management cluster.

To install a two data center disaster recovery deployment, you must set all of the placeholder properties in the CR file for both the Management and the Developer Portal subsystems as detailed in the relevant topics in the [Installing the API Connect subsystems](#) section, as well as the properties in the multi-site HA section that are detailed here. Some examples of complete CR files are shown in the [Example](#) section.

Procedure

• Converting a single data center deployment to two data centers

When converting a single data center deployment into a two data center deployment, it is best practice to use the existing data center as the active one, and add a new warm-standby data center elsewhere. This practice ensures that the data is retained.

Remember:

- Both data centers must have the same `portal-admin` and `portal-www` endpoints.
- You cannot configure a `siteName` for the existing Management or Portal cluster because the `siteName` property can be configured only at first deployment.

The following example shows how to update the multi-site HA CR values for making the local data center called `dallas` (DC1) active, and for setting the new remote data center called `raleigh` (DC2) as warm-standby.

• Install a cert-manager.

Use of a certificate manager adds convenience to the generation and management of certificate, but is not required. Whenever a custom resource (CR) takes a certificate secret name as input, you can point to any secret name, on condition that the secret exists before you deploy the CR, and that the secret contains relevant certificate data. Typically, this is `tls.crt`, `tls.key`, and `ca.crt` files. See [Certificates in a Kubernetes environment](#).

Note: Run these steps on DC1 only if you don't already have cert-manager running.

1. Obtain the certificate manager.

API Connect v10 uses cert-manager v1.9.1 of cert-manager, which is a native [Kubernetes](#) certificate management controller.

You can obtain cert-manager v1.9.1 from the API Connect v10 distribution `helper_files.zip` archive, or from <https://github.com/cert-manager/cert-manager/releases/tag/v1.9.1>.

Note: Cert-manager is bundled as a common service in IBM CloudPack for Integration.

2. Apply the CR:

```
kubectl apply -f cert-manager-1.9.1.yaml
```

Do not specify a custom namespace.

See <https://docs.cert-manager.io/en/release-0.10/getting-started/install/kubernetes.html>.

3. Wait for `cert-manager` pods to enter `Running 1/1` status before proceeding. To check the status:

```
kubectl get po -n cert-manager
```

There are 3 `cert-manager` pods in total.

- Use the following steps to allow `ingress-ca` secrets to be the same on both data centers.

1. On DC1 apply the file `ingress-issuer-v1-dc1.yaml`:

```
kubectl -n <namespace> apply -f ingress-issuer-v1-dc1.yaml
```

2. Validate that the command succeeded:

```
kubectl get certificates -n <namespace>
```

3. Export `ingress-ca` secret as a yaml from DC1:

```
kubectl -n <namespace> get secret ingress-ca -o yaml > ingress-ca.yaml
```

4. Edit the `ingress-ca.yaml` file to remove all labels, annotations, `creationTimestamp`, `resourceVersion`, `uid`, and `selfLink`.

5. Copy the `ingress-ca.yaml` from DC1 to DC2 and apply that file on DC2:

```
kubectl -n <namespace> apply -f ingress-ca.yaml
```

6. On DC2 apply the file `ingress-issuer-v1-dc2.yaml`:

```
kubectl -n <namespace> apply -f ingress-issuer-v1-dc2.yaml
```

7. Use the following commands to test that they are the same, on DC1 run:

```
kubectl -n <namespace> get secrets ingress-ca -o yaml | grep tls.crt | awk '{print $2}' | base64 -d > /tmp/ingress.pem.dc1
```

8. On DC2 run:

```
kubectl -n <namespace> get secrets ingress-ca -o yaml | grep tls.crt | awk '{print $2}' | base64 -d > /tmp/ingress.pem.dc2
```

9. To see the differences run:

```
diff /tmp/ingress.pem.dc1 /tmp/ingress.pem.dc2
```

The files should be the same.

10. To ensure that the certificates are working correctly and that they are using the `ingress-ca` secret. First, get the `portal-admin-client` crt file:

```
kubectl -n <namespace> get secrets portal-admin-client -o yaml | grep tls.crt | awk '{print $2}' | base64 -d > /tmp/admin-client.crt
```

11. Test that it is working by using OpenSSL:

```
openssl verify -verbose -CAfile /tmp/ingress.pem.dc1 /tmp/admin-client.crt
```

If it is working, you should see:

```
/tmp/admin-client.crt: OK
```

- **Installing to an active data center**

The following example shows how to set the multi-site HA CR values for deploying to an active data center (DC). In this instance, we are deploying to a local data center called `dallas` (DC1) that has a remote data center called `raleigh` (DC2).

1. Set Dallas to be active for the API Manager service on DC1.

As the API Manager service already exists, you cannot set the `siteName` property. Add the multi-site HA section properties to the existing `ManagementCluster` CR file for DC1. For example:

```
multiSiteHA:
  mode: active
  replicationEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: mgrreplicationdallas.cluster1.example.com
        secretName: dcl-mgmt-replication
      replicationPeerFQDN: mgrreplicationraleigh.cluster2.example.com
    tlsClient:
      secretName: dcl-mgmt-replication-client
```

2. Set Dallas to be active for the Developer Portal service on DC1.

As the Developer Portal service already exists, you cannot set the `siteName` property. Add the multi-site HA section properties to the existing `PortalCluster` CR file for DC1. For example:


```

multiSiteHA:
  mode: active
  replicationEndpoint:
    annotations:
      cert-manage.io/issuer: ingress-issuer
    hosts:
      - name: ptlreplicationdallas.cluster1.example.com
        secretName: dallas-ptl-replication-worker-1
      replicationPeerFQDN: ptlreplicationraleigh.cluster2.example.com
    tlsClient:
      secretName: ptl-replication-client

```

3. Create a secret called `ptl-encryption-key` on the active and warm-standby data centers. You must create a secret on the active and warm-standby sites that use the same random string, run the following command:

```
kubectl create secret generic ptl-encryption-key --from-literal=encryption_secret=<RANDOM STRING> -n <namespace>
```

The string can consist of uppercase letters, lowercase letters, and numbers, and must be at least 64 characters and no more than 100 characters.

4. When the secrets are created on both DC1 and DC2, update the Portal CR on DC1 and DC2 to include in the `spec` object:

```

encryptionSecret:
  secretName: ptl-encryption-key

```

Warning: Ensure that you add the secret to the `spec` object and not to the `multiSiteHA` object.

5. Apply the CR files to the subsystems in DC1.

For example, to apply the `ManagementCluster` CR file to the DC1 cluster, run the following command:

```
kubectl apply -f management_cr.yaml -n <namespace>
```

Where `<namespace>` is the target installation namespace in the Kubernetes cluster for DC1. Repeat this process to apply the updated `PortalCluster` CR file to DC1.

You can verify that the updated CR files have been applied by running the following command:

```
kubectl get ManagementCluster -n <namespace>
```

Change `ManagementCluster` to `PortalCluster` to verify the Developer Portal installation. For more information, see [Installing the API Connect subsystems](#).

6. Set your dynamic router to direct all traffic to DC1.

This includes setting the four endpoints for the API Manager cluster, and the two endpoints for the Developer Portal cluster.

- **Installing to a warm-standby data center**

The following example shows how to set the multi-site HA CR values for the remote data center called `raleigh` (DC2), which is being installed as the warm-standby, standby location. You must use the same encryption key on both sites.

1. Get the encryption key from Dallas the `active` (DC1) site:

- a. Run the command `kubectl get mgmt -o yaml | grep enc` to get the db encryption key on the `active` site, for example:

```
$ kubectl -n <namespace> get mgmt -o yaml | grep enc
```

An example of possible output:

```
encryptionSecret: dallas-enc-key
```

Make a note of the secret name. In this case, it's `dallas-enc-key`.

- b. Run `kubectl get secret dallas-enc-key -o yaml`, an example of possible output:

```

apiVersion: v1
data:
  encryption_secret.bin:
VKBNFj7sAOizxvE1H6i+9P31oJHvWwsO+x*****EGe/K+x6b3D7FEWgGoyG1WBUJKB4+T21My2iR5rBTov
pyLiY5g*****tSiRcQKegMPNBPGl829SVBCxuv3I=
kind: Secret
metadata:
  creationTimestamp: "2020-08-28T12:53:41Z"
  labels:
    app.kubernetes.io/instance: m1
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: dallas-enc-key
  name: dallas-enc-key
  namespace: default
  resourceVersion: "43039"
  selfLink: /api/v1/namespaces/default/secrets/m1-enc-key
  uid: 46c92395-9cc2-4421-b2c4-48e472c0cbb1
type: Opaque

```

- c. Copy the output that you got and put it into a new file. Delete the contents of `metadata:` and add back in a `name:` attribute, in this case `dc1-enc-key`. An example of the format is here:

```

apiVersion: v1
data:
  encryption_secret.bin:
VKBNFj7sAOizxvE1H6i+9P31oJHvWwsO+x*****EGe/K+x6b3D7FEWgGoyG1WBUJKB4+T21My2iR5rBTov
pyLiY5g*****tSiRcQKegMPNBPGl829SVBCxuv3I=
kind: Secret
metadata:
  name: dc1-enc-key
type: Opaque

```

- d. Save the file. In this example, it's called `dc1-enc-key.yaml`.

2. On Raleigh the (DC2) site:

- a. Copy the yaml file that you just saved that contains the `encryption_secret.bin` from Dallas the `active` (DC1) site.

- b. Run `kubectl create -f dc1-enc-key.yaml` to create the same secret with the same key on the Raleigh (DC2) site. For example:

```
$ kubectl -n <namespace> create -f /tmp/dc1-enc-key.yaml
```

An example of possible output:

```
secret/dc1-enc-key created
```

c. In the management CR for Raleigh the (DC2) site, add the following entry to the `spec`.

```
encryptionSecret:  
  secretName: dc1-enc-key
```

3. Set Raleigh to be warm-standby for the API Manager service on DC2.

Edit the `ManagementCluster` CR file `management_cr` for DC2, and set the multi-site HA section properties, for example:

```
multiSiteHA:  
  mode: passive  
  replicationEndpoint:  
    annotations:  
      cert-manage.io/issuer: ingress-issuer  
    hosts:  
      - name: mgrreplicationraleigh.cluster2.example.com  
        secretName: raleigh-mgr-replication-worker-1  
  replicationPeerFQDN: mgrreplicationdallas.cluster1.example.com  
  tlsClient:  
    secretName: mgr-replication-client  
  ...  
siteName: raleigh
```

4. Set Raleigh to be warm-standby for the Developer Portal service on DC2.

Edit the `PortalCluster` CR file `portal_cr` for DC2, and set the multi-site HA section properties, for example:

```
multiSiteHA:  
  mode: passive  
  replicationEndpoint:  
    annotations:  
      cert-manage.io/issuer: ingress-issuer  
    hosts:  
      - name: ptlreplicationraleigh.cluster2.example.com  
        secretName: raleigh-ptl-replication-worker-1  
  replicationPeerFQDN: ptlreplicationdallas.cluster1.example.com  
  tlsClient:  
    secretName: ptl-replication-client  
  ...  
siteName: raleigh
```

5. Ensure that you completed the steps that are in the **Installing to an active data center** section to create a secret called `ptl-encryption-key` on DC1 and DC2, and to update the Portal CR on DC1 and DC2 to include the `ptl-encryption-key` in the `spec` object.

6. Apply the CR files to the subsystems in DC2.

For example, to apply the `ManagementCluster` CR file to the DC2 cluster, run the following command:

```
kubectl apply -f management_cr.yaml -n <namespace>
```

Where `<namespace>` is the target installation namespace in the Kubernetes cluster for DC2. Repeat this process to apply the updated `PortalCluster` CR file to DC2.

You can verify that the updated CR files are applied by running the following command:

```
kubectl get ManagementCluster -n <namespace>
```

Change `ManagementCluster` to `PortalCluster` to verify the Developer Portal installation. For more information, see [Installing the API Connect subsystems](#).

Related concepts

- [Maintaining a two data center deployment](#)
- [Backup and restore requirements for a two data center deployment](#)
- [Recovering from a failover of a two data center deployment](#)
- [Removing a two data center deployment](#)

Related tasks

- [How to failover API Connect from the active to the warm-standby data center](#)

Deploying on OpenShift and Cloud Pak for Integration

You can install API Connect in an OpenShift environment or as the API Management capability in IBM Cloud Pak for Integration.

If you are deploying multiple API Connect instances in the same OpenShift cluster, all of them should be at the same version. The recommended configuration is to have a single operator in the global namespace, managing multiple API Connect instances in separate namespaces.

- [Preparing for installation](#)
Set up your OpenShift environment in preparation for deploying API Connect.
- [Installation procedures with IBM Cloud Pak for Integration](#)
The API Management capability of IBM Cloud Pak for Integration uses IBM API Connect.
- [Installing with the top-level CR on OpenShift](#)
Use the single, top-level `APIConnectCluster` CR to deploy API Connect on OpenShift.

- [Installing with subsystem CRs in a shared namespace on OpenShift](#)
Use the individual subsystem CRs to install API Connect subsystems in a shared namespace on OpenShift.
- [Installing with subsystem CRs in different namespaces or environments on OpenShift](#)
Install API Connect subsystems in different namespaces, for example Management subsystem in a namespace that is called `apicmgmt`, Gateway in a namespace called `apicgateway`. These same steps can also be used for installing subsystems in different environments.
- [Installing a two data center deployment with Cloud Pak for Integration](#)
How to deploy a two data center disaster recovery (2DCDR) deployment on OpenShift with Cloud Pak for Integration.
- [Installing a two data center deployment on OpenShift](#)
How to deploy a two data center disaster recovery (2DCDR) deployment on OpenShift.
- [Troubleshooting installation and upgrade on OpenShift](#)
Review the following troubleshooting tips if you encounter a problem while installing or upgrading API Connect on OpenShift, including as a component of IBM Cloud Pak for Integration (CP4I).

Preparing for installation

Set up your OpenShift environment in preparation for deploying API Connect.

About this task

Ensure that your environment meets all prerequisites, plan your deployment, and download files for installation.

Procedure

1. Ensure that your environment meets the following prerequisites:
 - [IBM API Connect Version 10 software product compatibility requirements](#)
Attention: API Connect is not supported on a FIPS-enabled environment.
 - [Deployment requirements](#)
 - If you want to use the OpenShift web console, Cluster admin access is required.
 - API Connect requires `block storage` as the storage class for all subsystems.
 - Set the default SCC (security context constraint) to `restricted`.
API Connect supports only the `restricted` SCC on OpenShift.
 - If your OpenShift deployment uses a `LimitRange` and a CPU `maxLimitRequestRatio` is set, then its value must be at least 25; otherwise some API Connect pods might fail to start.
2. Plan your deployment as explained in [Planning your deployment](#).
3. Review [Operator, operand, and CASE versions](#) to determine the operator, operand, and CASE versions that you need for your deployment.
4. Optional: Specify additional settings in each subsystem CR to define the deployment's configuration before beginning the installation.
In particular, consider increasing the timeout value for Management endpoints as explained in [Configuring timeouts for management endpoints](#).

For information on other configuration settings, see [API Connect configuration settings](#).

Tip: If you choose not to specify configuration settings now, you can modify the installation CRs later and then re-apply them to update the deployment.

- [Operator, operand, and CASE versions](#)
When you install API Connect on OpenShift, each release of API Connect supports a new combination of operator and operand, as well as a new CASE version for air-gapped installations.
- [API Connect configuration settings](#)
Specify configuration settings for API Connect in the CR, or the Platform UI in Cloud Pak for Integration.
- [Configuring timeouts for management endpoints](#)
Adjust the timeout settings for management endpoints by updating the CR.

Operator, operand, and CASE versions

When you install API Connect on OpenShift, each release of API Connect supports a new combination of operator and operand, as well as a new CASE version for air-gapped installations.

Table 1 lists the operator, operand, and CASE information that applies to each release of API Connect.

Table 1. Operators and operands for recent versions of API Connect

API Connect release	Operator channel	Operator version	CASE	DataPower Operator version
10.0.5.4	v3.4	3.4.0	4.0.5	1.6.8
10.0.5.3	v3.3	3.3.0	4.0.4	1.6.6
10.0.5.2	v3.2	3.2.0	4.0.2	1.6.3
10.0.5.1	v3.1	3.1.0	4.0.1	1.6.2
10.0.5.0	v3.0	3.0.0	4.0.0	1.6.0

API Connect configuration settings

Specify configuration settings for API Connect in the CR, or the Platform UI in Cloud Pak for Integration.

You can use the Platform UI on Cloud Pak for Integration, the OpenShift web console, or direct editing of the top-level or subsystem CR YAML files to specify advanced configuration options for API Connect. The following topics describe the available configuration options.

Attention: A mistake in a configuration update can result in an admission webhook error when `APIConnectCluster` is updated, or the `APIConnectCluster` can get stuck in a pending state with a webhook error message.

- Example of an admission webhook error when the `APIConnectCluster` update is rejected:
In this example, the user attempted to change the deployment profile to `n12xc4.m12-incorrect` and the result looked like the following example:

```
oc edit apiconnectcluster
```

```
error: apiconnectclusters.apiconnect.ibm.com "production" could not be patched: admission webhook "vapiconnectcluster.kb.io" denied the request: APIConnectCluster.apiconnect.ibm.com "production" is invalid: spec.Profile: Invalid value: "n12xc4.m12-incorrect": invalid profile. Options are: [n3xc4.m16 n12xc4.m12]
```

- Example of when the `APIConnectCluster` update is allowed, but is the state changes pending due to an underlying webhook error:

```
oc get apiconnectcluster -n apic
```

```
NAME   READY   STATUS              VERSION                               RECONCILED VERSION   AGE
m1     admission webhook "vmanagementcluster.kb.io" denied the request: ManagementCluster.management.apiconnect.ibm.com "m1-mgmt" is invalid: [spec.databaseBackup.restartDB.accept: Invalid value: false: databaseBackup protocol has changed (from '' to 'objstore') and requires database restart. accept restartDB to restart database, spec.databaseBackup.restartDB.accept: Invalid value: false: databaseBackup configuration has changed and requires database restart. accept restartDB to restart database] Pending 10.0.1.5-3388-eus 10.0.1.5-3388-eus 2d6h
```

Basic configuration

- [Basic configuration settings](#)

Advanced configuration

To enable access to the advanced configuration settings with Platform UI, toggle the Advanced Options slider.

- [Analytics subsystem settings](#)
- [Gateway subsystem settings](#)
- [Management subsystem settings](#)
- [Developer Portal subsystem settings](#)
- [Additional Kubernetes settings](#)

Additional configuration

You must edit the yaml file to set additional properties.

- [Basic configuration settings](#)
You can specify the API Connect basic configuration settings directly in the CR or with the Platform UI in Cloud Pak for Integration.
- [Management subsystem settings](#)
You can specify advanced configuration settings for the Management subsystem directly in the CR or with the Platform UI in Cloud Pak for Integration.
- [Gateway subsystem settings](#)
You can specify advanced configuration settings for the Gateway subsystem directly in the CR or with the Platform UI in Cloud Pak for Integration.
- [Developer Portal subsystem settings](#)
You can specify advanced configuration options for the Developer Portal subsystem directly in the CR or with the Platform UI in Cloud Pak for Integration.
- [Analytics subsystem settings](#)
Specify advanced configuration settings for the analytics subsystem directly in the analytics CR. If you are using Cloud Pak for Integration, you can configure many of the analytics setting in Platform UI.
- [Additional Kubernetes settings](#)
For each API Connect subsystem, you can specify additional settings for annotations and pod management.

Basic configuration settings

You can specify the API Connect basic configuration settings directly in the CR or with the Platform UI in Cloud Pak for Integration.

Review the tables for information on configuration settings for the following areas:

- [Details](#)
- [License](#)
- [Deployment profiles for installing with a single, top-level CR](#)
- [Product version](#)
- [Storage class](#)

Details

Table 1. Details

Field	Type	Required	Description
Name	String	Yes	Name this instance using only lowercase alphanumeric characters plus the underscore "-". Maximum of 54 characters. Default: none

License

Table 2. License

Field	Type	Required	Description
License acceptance	Boolean	Yes	The license agreement https://ibm.biz/apictoolkitlic must be accepted before you can install the API Connect capability. Default value: <code>false</code> (off). In the UI, click the License acceptance toggle to enable it. In the CR, set the <code>accept</code> : field to <code>true</code> to accept the license.
License use	String	Yes	Select production or nonproduction , to match the license you purchased. Default: nonproduction
License metric	Enumerated	No	Enter the unit of measure that is used for your program license: OpenShift <ul style="list-style-type: none"> PROCESSOR_VALUE_UNIT - Default value. If you leave the field blank, this value is used. MONTHLY_API_CALL - Applies only to the IBM API Connect Hybrid Entitlement program. Cloud Pak for Integration <ul style="list-style-type: none"> VIRTUAL_PROCESSOR_CORE - Default value. If you leave the field blank, this value is used. MONTHLY_API_CALL - Applies only to the IBM Cloud Pak for Integration - API Calls add-on program. For information on tracking monthly call volume, see Tracking API volume for auditing and compliance .
License ID	String	Yes	Select the license that you purchased for this instance of API Connect. Valid licenses are based on the selected value in the Product version field (see Table 4). When deploying the standalone IBM API Connect product, you must select the License ID for the API Connect program that you purchased. To view all License IDs, see API Connect licenses .

Deployment profiles for installing with a single, top-level CR

Select a deployment profile for the new installation. See [API Connect deployment profiles for OpenShift and Cloud Pak for Integration](#) for more information on the available profiles.

For information on switching to a different deployment profile after installation, see the following topics:

- [Changing the deployment profile on Cloud Pak for Integration](#)
- [Changing deployment profiles on OpenShift](#)

Product version

Table 3. Product version

Field	Type	Required	Description
Product version	Enumerated	No	Select the product version to be installed in the system. Warning: Do not select a channel here instead of the API Connect version. Selecting a channel will result in any API Connect interim fix updates being applied automatically and without warning.

Storage class

Table 4. Storage class

Field	Type	Required	Description
Default Storage Class	Enumerated	No	Specify the RWO block storage class to use for persistence storage. This will be used when creating a PVC. Specification of the storage class is optional, but is highly recommended. If not specified the default cluster storage class is used. See the IBM Cloud Pak documentation for guidance for Storage class selection. Go to the following page and select your Cloud Pak release level: https://www.ibm.com/support/knowledgecenter/SSGT7J . In the navigation panel, select System Requirements > Storage. To review API Connect storage support, see Storage classes .

Management subsystem settings

You can specify advanced configuration settings for the Management subsystem directly in the CR or with the Platform UI in Cloud Pak for Integration.

The IBM Cloud Pak Platform UI enables you to specify settings for the following areas:

- [Database backup](#)
- [Database persistence](#)
- [Site name](#)
- [Test and Monitor](#)
- [Billing](#)

For annotations, labels, node selectors, and tolerations, see [Additional Kubernetes settings](#).

Note: Optionally, you might want to increase the timeout settings for management endpoints, particularly for large deployments. See [Configuring timeouts for management endpoints](#).

Database backup

Table 1. Database backup

Field	Type	Required	Description
Credentials	Drop down menu	No. Optional	<ul style="list-style-type: none"> For objstore type backups, the name of the Kubernetes secret containing your S3 Access Key / Key Secret For SFTP type backups, the name of the Kubernetes secret containing your SFTP Username/password <p>Default: (Generated Value)</p> <p>See Configuring backup settings for fresh install of the Management subsystem.</p>
Server hostname	String	No. Optional	<p>The backups host:</p> <ul style="list-style-type: none"> For objstore type backup, specify S3 endpoint with the corresponding S3 region in the format <S3endpoint>/<S3region>. For example: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard For sftp type, the SFTP server hostname. For example: 203.0.113.0 or example.com Not required for local storage backups <p>Default: none</p> <p>See Configuring backup settings for fresh install of the Management subsystem.</p>
Path	String	No. Optional	<p>The path to the location of the backup:</p> <ul style="list-style-type: none"> For objstore backups, the name of your S3 bucket to store backup data. For example, bucket-name/folder. The use of subdirectories in the bucket name is not supported. Ensure that bucket-name/folder is empty. If the folder was previously used for backups, the folder will not be empty, and stanza create might encounter an error. For sftp type, the folder name on the SFTP server. For example: mgmt_backup. Not required for local storage backups <p>Default: none</p> <p>See Configuring backup settings for fresh install of the Management subsystem.</p>
Server port	String	No. Optional	<p>Backup server port</p> <p>The port for the protocol to connect to the host. The backup port is not required for object storage. Default: 22</p> <p>See Configuring backup settings for fresh install of the Management subsystem.</p>
Protocol	String	No	<p>The type of the backup. Supported types:</p> <ul style="list-style-type: none"> objstore sftp The default value is sftp. Local storage backups For local backups, do not specify this parameter. When the parameter is set to "", the backup type is set to local storage backup. <p>See Configuring backup settings for fresh install of the Management subsystem.</p>
Restart database	Toggle Switch	No. Optional	<p>Flag to accept database reconfiguration. Restarting the database is required when changing the backup configuration. When marked true and the new backup configuration differs, the operator will stop the database and any dependent services, apply the change, and restart the stopped components.</p> <p>Defaults to Off.</p> <p>For more information on restarting the database, see Reconfiguring or adding backup settings after installation of the management subsystem</p>
Retries	String	No. Optional	<p>For sftp type backups only, the number of times the ibm-apiconnect Operator attempts backups in the event of a failed SFTP backup.</p> <p>Not used for objstore or local storage backup.</p> <p>Default value: 0.</p> <p>See Configuring backup settings for fresh install of the Management subsystem.</p>
S3 provider type	Drop down menu	Required when using s3 provider.	<p>Name of the S3 provider to use. Not required for local backups or SFTP backups.</p> <p>If you are configuring backups with an s3provider, you must specify one of the supported values of aws or ibm.</p> <p>See Configuring backup settings for fresh install of the Management subsystem.</p>

Field	Type	Required	Description
Schedule	String	No. Optional	<p>Cron like schedule for performing automatic backups. The format for the schedule is:</p> <ul style="list-style-type: none"> • * * * * * • - - - - - • • +----- day of week (0 - 6) (Sunday=0) • +----- month (1 - 12) • +----- day of month (1 - 31) • +----- hour (0 - 23) • +----- min (0 - 59) <p>The backup schedule defaults to 0 0 * * *. This means a backup is run every day at midnight and minute zero UTC.</p> <p>When you configure a host, if you do not specify a value for <code>schedule</code>, the default backup schedule is automatically set. Note that the default backup schedule is not set, and scheduled backups not enabled, until host configuration is completed.</p> <p>See Configuring backup settings for fresh install of the Management subsystem.</p>

Database persistence

Table 2. Database persistence

Field	Type	Required	Description
Storage Class	Drop down menu	No. Optional	<p>Specify the RWO block storage class to use for persistence storage. This will be used when creating a PVC. Specification of the storage class is highly recommended. If not specified the default cluster storage class is used.</p> <p>See the IBM Cloud Pak documentation for guidance for Storage class selection. Go to the following page and select your Cloud Pak release level: https://www.ibm.com/support/knowledgecenter/SSGT7J. In the navigation panel of the KC, select System Requirements > Storage.</p> <p>To review API Connect storage support, see Storage classes.</p>
Volume size	Integer	No. Optional	<p>Represents the size of the Database volume. Defaults to 120G for <code>n1xc4.m16</code> installation profile, 180G for <code>n3xc4.m16</code> installation profile.</p>

Site name

Table 3. Site Name

Field	Type	Required	Description
Site name	string	No. Optional	<p>Site name is the logical name of this failure domain. Typically site name is the name of your data center, such as Amazon AZ.</p> <p>Default: randomly generated name.</p> <p>This will be the site name of the management deployment. For example, given the site name of <code>test</code> your management deployment will look like <code>management-test-postgres</code>.</p>

Test and Monitor

Table 4. Test and Monitor

Field	Type	Required	Description
Enable Test and Monitor	Boolean. Toggle off/on	No. Optional	<p>Test and Monitor automatically generates API test assertions. It contains dashboards for monitoring API test assertions and getting alerts on the health of your APIs.</p> <p>Default: off</p> <p>See Installing the Automated API behavior testing application.</p>

Billing

Table 5. Billing

Field	Type	Required	Description
Enable Billing	Boolean. Toggle off/on	No. Optional	<p>To enable the monetization of your Product Plans, you must add a billing integration resource in your API Connect provider organization that defines the configuration data needed to synchronize with an external subscription billing system.</p> <p>Default: off</p> <p>See Adding a billing integration resource.</p>

Gateway subsystem settings

You can specify advanced configuration settings for the Gateway subsystem directly in the CR or with the Platform UI in Cloud Pak for Integration.

The IBM Cloud Pak Platform UI enables you to specify settings for the following areas:

- [v5 compatibility mode](#)
- [Operations Dashboard](#)
- [Token management service](#)
- [Autoscaling gateway pods](#)

For annotations, labels, node selectors, and tolerances, see [Additional Kubernetes settings](#)

v5 compatibility mode

Table 1. v5 compatibility mode

Field	Type	Required	Description
V5 comparability mode	Toggle Switch off/on	No. Optional	Enabling this provides v5 compatibility (trading off some performance) with the IBM DataPower Gateway that was provided with IBM API Connect Version 5 and earlier releases. Consider using this DataPower Gateway (v5 compatible) if you are an existing DataPower user and want to utilize your DataPower resources and knowledge. Default: off To review v5 compatibility, see API Connect gateway types .

Operations Dashboard

Table 2. Operations Dashboard

Field	Type	Required	Description
Enable Operations Dashboard tracing	Toggle Switch off/on	No. Optional	Enable or disable cross-component transaction tracing to allow troubleshooting and investigating errors and latency issues. Default: disabled (off) To review the Operations Dashboard, see https://www.ibm.com/support/knowledgecenter/SSGT7J_20.3/tracing/operations_dashboard.html For information on configuring the Operations Dashboard, see https://ibm.github.io/datapower-operator-doc/apis/datapowerservice/v1beta2/#odtracing .
Operations Dashboard namespace	string	No. Optional	Namespace where the IBM Cloud Pak for Integration Operations Dashboard has been deployed. Default: none
Replica count	string	No. Optional	Specify the number of instances of the Gateway that should be deployed. For high-availability, a minimum of 3 replicas (one per node) is recommended. Default: none

Token management service

Table 3. Token management service

Field	Type	Required	Description
Advanced: Enable TMS	Toggle Switch	No. Optional	Enable or disable token management service (TMS) to use the DataPower to manage the token lifecycle.. Default: disabled (off) The Token Management Service uses the DataPower distributed cache to manage the token lifecycle which includes when to revoke access rights. For more information, see Token management with DataPower API Gateway .
Advanced: Storage Class	string	No. Optional	Specify the RWO block storage class to use for persistence storage. This will be used when creating a PVC. Specification of the storage class is highly recommended. If not specified the default cluster storage class is used. See the IBM Cloud Pak documentation for guidance for Storage class selection. Go to the following page and select your Cloud Pak release level: https://www.ibm.com/support/knowledgecenter/SSGT7J . In the navigation panel of the KC, select System Requirements > Storage. To review DataPower storage configuration, see https://ibm.github.io/datapower-operator-doc/apis/datapowerservice/v1beta2/#storage .
Advanced: Volume size	string	No. Optional	Resources represents the minimum resources the volume should have. For example, 10Gi. Default: none To review DataPower storage configuration, see storage API . in the DataPower documentation.

Autoscaling gateway pods

Autoscaling enables the gateway deployment to scale dynamically, either vertically or horizontally. It is not possible to use both horizontal and vertical pod scaling simultaneously. You must configure either **HPA** (for horizontal scaling) or **VPA** (for vertical scaling) as the scaling method.

Note: DataPower Gateway supports long-lived connections such as GraphQL subscriptions or other websockets connections. These long-lived connections might not be preserved when pods are scaled down based on defined CPU or memory thresholds. Workloads with long-lived connections are more vulnerable to failed API transactions when auto-scaling is occurring.

To learn more about HPA and VPA autoscaling, see the following topics in the DataPower documentation:

- [Autoscaling DataPower Pods](#)
- [podAutoScaling API](#)

You can configure autoscaling for Gateway pods by inserting a new subsection in the **spec gateway** section of the top-level CR, or by filling in the corresponding fields in the OpenShift or IBM Cloud Pak Platform UI. Settings are described in Table 4.

The following example configures HPA autoscaling in the gateway section of the top-level API Connect CR:

```
spec:
  gateway:
    podAutoScaling:
      hpa:
        maxReplicas: 5
        minReplicas: 3
        targetCPUUtilizationPercentage: 90
        targetMemoryUtilizationPercentage: 80
      method: HPA
```

Table 4. Autoscaling configuration settings

UI Field name / CR Setting	Autoscaling method	Required	Description
<ul style="list-style-type: none"> Scaling method <code>method</code> 	All	Yes	Select the autoscaling method to be used: either HPA (Horizontal Pod Autoscaling) or VPA (Vertical Pod Autoscaling). To disable autoscaling, leave <code>method</code> blank, or omit the <code>podAutoScaling</code> section from the CR.
<ul style="list-style-type: none"> Minimum pods <code>minReplicas</code> 	HPA	Yes	Minimum number of pods that can be scaled down to by the horizontal pod autoscaler.
<ul style="list-style-type: none"> Maximum pods <code>maxReplicas</code> 	HPA	Yes	Maximum number of pods that can be scaled up to by the horizontal pod autoscaler. Must be a value greater than <code>minReplicas</code> .
<ul style="list-style-type: none"> Target CPU Utilization Percentage <code>targetCPUUtilizationPercentage</code> 	HPA	No	Percent threshold on average CPU usage by pods for triggering horizontal scaling.
<ul style="list-style-type: none"> Target Memory Utilization Percentage <code>targetMemoryUtilizationPercentage</code> 	HPA	No	Percent threshold on average memory usage by pods for triggering horizontal scaling.
<ul style="list-style-type: none"> Maximum CPU <code>maxAllowedCPU</code> 	VPA	No	Maximum value for CPU resource that can be assigned to a DataPower container when scaling vertically. Minimum value is set from <code>DataPowerRequestsSpec.ResourceCPU</code>
<ul style="list-style-type: none"> Maximum Memory <code>maxAllowedMemory</code> 	VPA	No	Maximum amount of memory that can be assigned to a DataPower container when scaling vertically. Minimum value is set from <code>DataPowerRequestsSpec.ResourceMemory</code>

Developer Portal subsystem settings

You can specify advanced configuration options for the Developer Portal subsystem directly in the CR or with the Platform UI in Cloud Pak for Integration.

The IBM Cloud Pak Platform UI enables you to specify settings for the following areas:

- [Database backups](#)
- [Site name](#)

For annotations, labels, node selectors, and tolerations, see [Additional Kubernetes settings](#)

Known limitation: When deployed on OpenShift or Cloud Pak for Integration, the API Connect Developer Portal subsystem does not support multiple hosts for the `portalUIEndpoint` endpoint, due to a limitation of Routes in OCP.

Database backups

Table 1. Database backup settings

Field	Type	Description
Credentials	String	For <code>objstore</code> , this is the name of your Kubernetes secret that was generated with your username and password for your backup database. For <code>sftp</code> , this is the secret that contains the credentials for your <code>sftp</code> username and password. Required if configuring backups. See also Backing up and restoring the Developer Portal in a Kubernetes environment .
Backup host	String	The S3 endpoint with the corresponding S3 region in the format <code><S3endpoint>/<S3region></code> . For example: <code>s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard</code> For <code>sftp</code> , this is the <code>sftp</code> server host name. Required if configuring backups. See also Backing up and restoring the Developer Portal in a Kubernetes environment .
Backup path	String	For <code>objstore</code> , the name of your S3 bucket to store backup data. For example: <code>test-bucket/restore-test</code> . For <code>sftp</code> , this is the path to the folder on the <code>sftp</code> server. Required if configuring backups. See also Backing up and restoring the Developer Portal in a Kubernetes environment .
Backup port	Integer	This is the port for the protocol to connect to the host. There is no default. See also Backing up and restoring the Developer Portal in a Kubernetes environment .

Field	Type	Description
Backup protocol	String	Choose the protocol (sftp or objstore) you wish to use to communicate with your remote backup endpoint. Objstore is recommended for most reliability. There is no default. This field must be set to either <code>sftp</code> or <code>objstore</code> to configure backups. See also Backing up and restoring the Developer Portal in a Kubernetes environment .
Backup schedule	String	This is the schedule in cron string format for how often automatic Developer Portal backups are run. For example: <code>0 /3 * * *</code> means at minute 0 past every 3rd hour. There is no default. Required if configuring backups. See also Backing up and restoring the Developer Portal in a Kubernetes environment .

Site name

Table 2. Site name

Field	Type	Description
Site name	string	Site name is the logical name of the failure domain of this portal subsystem. Typically site name is the name of your data center, such as Amazon AZ. The default is a randomly generated name.

Analytics subsystem settings

Specify advanced configuration settings for the analytics subsystem directly in the analytics CR. If you are using Cloud Pak for Integration, you can configure many of the analytics setting in Platform UI.

In the Cloud Pak for Integration Platform UI you can specify settings for the following areas:

- [Backups](#)
- [Storage type, class, and size](#)

For annotations, labels, node selectors, and tolerations, see [Additional Kubernetes settings](#)

Analytics features such as offloading, filtering, and disabling ingestion, must be configured directly in the analytics CR YAML file: [Analytics subsystem CR configured settings](#).

Backups

Table 1. Database backup

Field	Type	Required	Description
Backup chunk size in GB	Integer	No. Optional	<code>chunkSize</code> specifies how large files are stored. Large files can be stored as chunks when the snapshot is created. This setting specifies the size of the chunks as GB, MB, or KB. Default value is 1 GB. See Configuring backup settings for Analytics .
Credentials	String	No. Optional	The backup credential is the name of the Kubernetes secret that you created for accessing your S3-compatible storage. No default. See Configuring backup settings for Analytics .
Compress	Boolean: off or on	No. Optional	Determines whether metadata files are stored in compressed format. Default value is false (off). See Configuring backup settings for Analytics .
Enable server-side encryption	Boolean: off or on	No. Optional	Determines whether files are encrypted. When set to true (on), files are encrypted on the server side by using AES256. Default is false (off). See Configuring backup settings for Analytics .
Analytics server hostname	String	No. Optional	The URL to your S3 storage. No default. See Configuring backup settings for Analytics .
Backup Path	String	No. Optional	The path to your S3 storage at the URL specified by Analytics server hostname.host. No default. See Configuring backup settings for Analytics .
Schedule cron expression	String	No. Optional	The schedule that determines how often backups are run automatically. The format for the schedule is any valid <code>cron</code> string. Default: <code>"*/20 * * * *"</code> See Configuring backup settings for Analytics .

Storage type, class, and size

Table 2. Storage type, class, and size settings

Field	Type	Required	Description
Master storage class	String	Yes	If you specified dedicated storage, the master storage class is the storage class that is used by your <code>storage-os-master</code> pods. Select from the drop down. Local-volume storage is recommended for best performance. If none selected, the default storage class of your cluster is used. See Planning the analytics profile, storage class, and storage type .

Field	Type	Required	Description
Shared storage class	String	Yes	The storage class used by your analytics <code>storage</code> pods. Select from the drop down. Local-volume storage is recommended for best performance. If none selected, the default storage class of your cluster is used. See Planning the analytics profile, storage class, and storage type .
Master volume size	Integer > 0	No	If you specified dedicated storage, then master volume size specifies the disk storage that is allocated to your <code>storage-os-master</code> pods. The <code>storage-os-master</code> pods do not store your analytics data, and if left unset the value defaults to 5Gi.
Shared volume size	Integer > 0	Yes	How much persistent disk storage is allocated for analytics data (in Gi). The default is 50Gi, but it is HIGHLY recommended that you calculate how much storage you need. See Estimating internal storage space
Type	Boolean: dedicated or shared	Yes	OpenSearch storage type to use for your analytics data. For more information about storage types, see Dedicated storage instead of shared storage . The default storage type is shared.

- [Analytics subsystem CR configured settings](#)
Update the OpenShift analytics or top-level CR to configure additional analytics features.

Analytics subsystem CR configured settings

Update the OpenShift analytics or top-level CR to configure additional analytics features.

About this task

Based on the decisions you made while [defining your topology](#), configure your deployment options by completing the steps in the following topics to enable the persistent queue, configure data offloading, use dedicated storage, and disable internal storage.

Important: All additional analytics configuration options can be added, changed, or removed after installation **except** for disablement or enablement of internal storage. If you want to disable internal storage you must configure this in the analytics CR before installation: [Disabling internal storage](#)

- [Dedicated storage instead of shared storage](#)
Use dedicated storage instead of shared storage for your analytics data.
- [Analytics persistent queue](#)
For enhanced reliability in your analytics data pipeline, enable the persistent queue feature by updating your OpenShift analytics or top-level CR.
- [Offloading data to a third-party system](#)
Update the OpenShift analytics or top-level CR to configure data offloading from your IBM API Connect analytics deployment to a third-party system.
- [Disabling internal storage](#)
Update the OpenShift top-level or analytics CR to disable local storage in your IBM API Connect analytics deployment.
- [Modifying incoming analytics data](#)
Update the OpenShift top-level or analytics CR to customize data in your analytics deployment.
- [Reference: Fields in the Analytics CR](#)
Use the table of analytics-specific fields to configure the top-level or analytics CR for installing your API Connect analytics subsystem.

Dedicated storage instead of shared storage

Use dedicated storage instead of shared storage for your analytics data.

API Connect provides two OpenSearch storage types:

- **Shared storage**
A single pod provides data storage and OpenSearch cluster management. Shared storage is the default option, and the only option on a one replica deployment.
- **Dedicated storage**
OpenSearch data storage and cluster management functions are split, and run in separate pods on each worker node:
 - `storage-os-master` - Runs an OpenSearch "manager eligible" node that manages the OpenSearch cluster, but does not store any API event data.
 - `storage` - Runs an OpenSearch "data" node that stores API event data.

Dedicated storage facilitates horizontal scaling to support greater analytics data storage and throughput. Add worker nodes and increase the number of `storage` pod replicas in order to scale horizontally (you do not need to increase the number of `storage-os-master` replicas). Dedicated storage also provides greater stability and allows some OpenSearch configuration changes to be made without downtime.

The storage type is defined in the analytics CR, in the `spec.storage.type` section as follows:

- Shared storage:

```
storage:
  type: shared
  shared:
    volumeClaimTemplate:
      storageClassName: $STORAGE_CLASS
      volumeSize: $DATA_VOLUME_SIZE
```

- Dedicated storage:

```
storage:
  type: dedicated
  shared:
    volumeClaimTemplate:
      storageClassName: $STORAGE_CLASS
      volumeSize: $DATA_VOLUME_SIZE
  master:
```

```
volumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
```

Analytics persistent queue

For enhanced reliability in your analytics data pipeline, enable the persistent queue feature by updating your OpenShift analytics or top-level CR.

The Logstash persistent queue is an optional feature that provides enhanced reliability in the data pipeline. With the persistent queue enabled, the ingestion pod buffers incoming API event data to persistent storage so that no events are lost if the downstream API Connect services become unavailable.

For example, if the analytics storage service is restarted, any incoming API events are kept in persistent storage and sent on to the storage service when it is available again. If the persistent queue is not enabled, then the ingestion pod rejects further API event data coming from the gateway.

Another benefit of this feature is better handling of unexpected traffic spikes. The only disadvantages are higher CPU and persistent storage requirements.

To enable the persistent queue feature:

1. If using a top-level CR edit your **APIConnectCluster** CR file. Otherwise edit your `analytics_cr.yaml` file and add the following to the ingestion section

```
2. ingestion:
  queue:
    type: persisted
    volumeClaimTemplate:
      storageClassName: <storage-class-name>
      volumeSize: <storage size>
```

Where

- `<storage-class-name>` is the OpenShift storage class defined for persistent volumes. For example `local-storage`
 - `<storage size>` is the maximum size for your persistent queue. For example `8Gi`.
3. Save the file. If updating these settings after installation, confirm that the analytics ingestion pod restarts.

Offloading data to a third-party system

Update the OpenShift analytics or top-level CR to configure data offloading from your IBM® API Connect analytics deployment to a third-party system.

About this task

For information about offloading data from your Analytics deployment, see [Planning to offload data to a third-party system](#).

Procedure

1. If using a top-level CR edit your **APIConnectCluster** CR file and make the changes in the `spec.analytics` section. Otherwise edit your `analytics_cr.yaml` file and make changes in the `spec` section.
- 2.

```
external:
  offload:
    enabled: true
    output: |
      $EXTERNAL_OFFLOAD_OUTPUT
```

where `$EXTERNAL_OFFLOAD_OUTPUT` is the plugin output for the target third-party system.

The following example enables offloading to HTTP:

```
external:
  offload:
    enabled: true
    output: |
      http {
        url => "example.com"
        http_method => "post"
        codec => "json"
        content_type => "application/json"
        id => "offload_http"
      }
```

For examples of output plugins, see: [Sample output plugins for offloading data](#).

3. Optional: Modify the output plugin to include certificates for connecting to a secure endpoint. If your third-party endpoint requires a particular certificate, you can configure the certificate and then reference it in the offload plugin. Include the `secretName` as well as the `cacert`, `client_cert`, and `client_key` settings.

```
external:
  offload:
    secretName: offload-certificates
    enabled: true
    output: |
      http {
        url => "https://example.com:443"
```

```

http_method => "post"
codec => "json"
content_type => "application/json"
id => "offload_http"
cacert => "/etc/velox/external_certs/offload/cacert.pem"
client_cert => "/etc/velox/external_certs/offload/client.pem"
client_key => "/etc/velox/external_certs/offload/client.key"
}

```

The secret and its defined files are mounted to the `/etc/velox/external_certs/offload` directory in the ingestion pod container.

For example, the following output plugin enables offloading to HTTP, and provides information on the certificate that is needed for authenticating with the endpoint:

For information on creating a secret to store the certificates, see [Providing a custom certificate for Analytics offload](#).

- Optional: Add a **filter** after the output plugin to modify data prior to offloading it.

The following example filter removes the `query_string` field:

```

external:
  offload:
    enabled: true
    output: |
      http {
        url => "example.com"
        http_method => "post"
        codec => "json"
        content_type => "application/json"
        id => "offload_http"
      }
    filter: |
      ruby {
        remove_field => ["query_string"]
      }
}

```

For information about modifying offloaded data, see [Planning to modify your incoming analytics data](#).

- Save the file. If updating these settings after installation, confirm that the analytics ingestion pod restarts.

- [Sample output plugins for offloading data](#)

Review sample output plugins to see how to code your own plugins for offloading data from IBM API Connect Analytics.

- [Providing a custom certificate for Analytics offload](#)

Prepare a private or self-signed certificate for offloading data and then add it to the IBM API Connect Analytics deployment on OpenShift.

Sample output plugins for offloading data

Review sample output plugins to see how to code your own plugins for offloading data from IBM® API Connect Analytics.

Configure output plugins that direct analytics data to your third-party systems. An output plugin specifies where analytics data is offloaded, plus any information required to access that system, such as certificate information. The output plugin for displaying analytics data in the Analytics user interface is configured by default.

Note: If your third-party system requires a private or self-signed certificate, configure it as explained in [Providing a custom certificate for Analytics offload](#) and include a reference to the certificate in your output plugin. See the second HTTP output plugin for an example of how to reference the certificate.

HTTP

To configure data offload to an HTTP server, you must provide the server URL. You can optionally add standardized or custom HTTP headers to define additional operating parameters.

The following example configures an HTTP output plugin.

```

http {
  url => "example.com"
  http_method => "post"
  codec => "json"
  content_type => "application/json"
  headers => {
    "x-acme-index" => "8"
    "x-acme-key" => "0d5d259f-b8c5-4398-9e58-77b05be67037"
  }
  id => "offload_http"
}

```

The following example configures an HTTP output plugin that references a custom certificate for accessing the third-party endpoint.

```

http {
  url => "https://example.com:443"
  http_method => "post"
  codec => "json"
  content_type => "application/json"
  headers => {
    "x-acme-index" => "8"
    "x-acme-key" => "0d5d259f-b8c5-4398-9e58-77b05be67037"
  }
  id => "offload_http"
  cacert => "/etc/velox/external_certs/offload/cacert.pem"
  client_cert => "/etc/velox/external_certs/offload/client.pem"
  client_key => "/etc/velox/external_certs/offload/client.key"
}

```

For more information on offloading to HTTP, refer to the Logstash documentation on configuring the [HTTP output plugin](#).

Elasticsearch

To configure data offload to an Elasticsearch cluster, you must provide one or more server URLs, define how indices should be created by specifying a dynamically configured name, specify the number of primary shards for storing the indexed documents, and specify a number of replica shards for redundancy. You can optionally specify server authentication credentials.

The naming syntax that you specify for your indices determines how many indices are created and how they are characterized. For example, you can define indices that are created based on a date pattern (for example, daily or weekly), or indices that are created to store data by provider organization or by API name.

The following example configures an Elasticsearch output plugin.

```
elasticsearch {
  hosts => "https://example.com:443"
  index => "apiconnect"
  ssl => true
  cacert => "/etc/pki/tls/cert.pem"
}
```

For more information on offloading to Elasticsearch, refer to the Elasticsearch documentation on the [Elasticsearch output plugin](#).

Apache Kafka

To configure data offload to a Kafka cluster, you must provide host and port connection details for one or more servers, and the name of the Kafka topic to which you want to publish the offloaded data. For logging and monitoring purposes within Kafka, you can optionally specify a string identifier by which API Connect can be uniquely identified.

The following example configures a Kafka output plugin.

```
kafka {
  topic_id => "test"
  bootstrap_servers => "example.com:9093"
  codec => "json"
  id => "kafka_offload"
  ssl_truststore_location => "/usr/share/logstash/jdk/lib/security/cacerts"
  ssl_truststore_password => "changeit"
  ssl_truststore_type => "JKS"
  security_protocol => SSL
}
```

For more information on offloading to Apache Kafka, refer to the Logstash documentation on the [Kafka output plugin](#).

Syslog

To configure data offload to Syslog, you must provide host and port connection details for the server.

Restriction: Syslog does not support the use of chained client certificates for TLS profiles.

The following example configures a Syslog output plugin.

```
syslog {
  host => "example.com"
  port => 601
  protocol => "ssl-tcp"
  id => "offload_syslog"
  appname => "apiconnect"
  msgid => "%{org_id}"
  facility => "log audit"
  severity => "informational"
  codec => "json"
  ssl_cacert => "/etc/pki/tls/cert.pem"
}
```

Attention: Due to a third-party issue with Logstash, most of the facility values are not set correctly on the offload server. As a temporary workaround, set the facility to "user-level" to ensure that the corresponding value is correctly sent to the offload server. When Logstash corrects this problem, you must update the offload configuration to change the value to a more appropriate setting.

For more information on offloading to Syslog, refer to the Logstash documentation on the [Syslog output plugin](#).

OpenSearch

To configure data offload to OpenSearch: <https://opensearch.org/docs/latest/clients/logstash/ship-to-opensearch/>

Providing a custom certificate for Analytics offload

Prepare a private or self-signed certificate for offloading data and then add it to the IBM® API Connect Analytics deployment on OpenShift.

Procedure

1. Log in to a server where you can run `oc` commands against the OpenShift environment where your analytics subsystem is installed.
2. Encode the certificate file contents in base64.
You can use either JKS files or text-based files such as PEM files:

- If you use a JKS file for the certificate, you must encode it.
- If you use a text-based file such as PEM, then you can either encode it, or paste its contents directly.

There are various tools for base64 encoding. If the `cat` and `base64` commands are available you can run the following command to encode the file contents:

```
cat keystore.jks | base64
```

Copy the output so that you can paste it into the secret in the next step.

3. Create a secret to contain the certificate.

- Create a file called `analytics_certs.yaml` to contain the secret.

The secret's name that you specify in the file does not have to match the file name. In the examples, the secret's name is `offload-certificates`. Do not use the underscore character (`_`) in the secret name.

The method for adding the certificate to the secret depends on whether you encoded the file contents in the previous step:

- Base64-encoded file: In the `data` section of the secret, type a name for the field (you can choose the name). Paste the encoded output from the previous step (enclosed in `" "`) as the value.

```
apiVersion: v1
kind: Secret
metadata:
  # Change value of name to be whatever you wish secret to be called
  name: offload-certificates
  # Only base64 encoded data should be placed in data section; JKS keys would go here
data:
  keystore.jks: "output_of_base64_encoded_jks"
```

- Plain text file: In the `stringData` section of the secret, type a name for the key or certificate (you can choose the name), and paste the value.

```
apiVersion: v1
kind: Secret
metadata:
  # Change value of name to be whatever you wish secret to be called
  name: offload-certificates
stringData:
  cacert.pem: |-
-----BEGIN CERTIFICATE-----
MIIDIDCCAomgAwIBAgIEND70zzANBqkqhkiG9w0BAQUFADBOMQswCQYDVQGEwJVUzEQMA4GA1UE
ChMHRXFlaWZheDEtMCsGA1UECXMkRXFlaWZheCBTZWN1cmUgQ2VydG1maWNhdGUGUgQXV0aG9yaXR5
MB4XDTE4MDgyMjE2NDU1MVoXDTE4MDgyMjE2NDU1MVoVOTJELMakGAlUEBhMCVVMxEDA0BgNVBAoT
B0VxdWlmYXgkLTArBgNVBAsTJEVxdWlmYXgkU2VjdXJlIEN1cnRpZmljYXRlIEF1dGhvcml0eTCB
nzANBqkqhkiG9w0BAQEFAAOBjQAwGgYKCoGgYEAwV2xWGCiYu6gmi0fCG2RFGiYCh7+2gRvE4RiIcPR
fm6fBeC4AfbONOziipUEZKzxa1NfBbPLZ4C/QgKO/t0BCezhABRP/PvwdN1Dulsr4R+AcJkVv5MW
8Q+XarfCaMcZelZMKxRHjuvK9buY0V7xd1fUNLjUA86iOe/FP3gx7kCAwEAaOACAQkwggEFMHAG
AlUdHwRpmGcwZaBjoGGkXzBdMQswCQYDVQGEwJVUzEQMA4GA1UEChMHRXFlaWZheDEtMCsGA1UE
CxMkRXFlaWZheCBTZWN1cmUgQ2VydG1maWNhdGUGUgQXV0aG9yaXR5MQ0wCwYDVQQDEwRDUkwxMBoG
AlUdEAQTMGBDZlIwMTgwODIyMTY0MTUxWjALBgNVHQ8EBAMCAQYwHwYDVR0jBBgwFoAUSOZo+Svs
spXXR9gjIBBPM5iQn9QwHQYDVR0OBBYEFjmaPkr0rKV10fYIyAQZtOYkJ/UMAwGAlUdEwQFMAMB
Af8wGgYJKoZIhVZ9B0EABA0wCxsFVjMuMGMDAgbAMA0GCSqGSIb3DQEBBQUAA4GBAFjOKer89961
zgk5F7WF0bnj4JXMJTENAKaSbn+2kmOeUJXRmm/kEd5jhW6Y7qj/WsjTVbJmcVfewCHRPSqnI0kB
BIZCe/zuf6IWURVnZ9NA2zsmWLIodz2uFHdh1voqZiegDfqnclzqcPGUIWVEX/r87yloqaKHee95
70+sB3c4
-----END CERTIFICATE-----
```

You can add multiple keys/certificates, setting the field names for each as desired. For each key, a file based on the corresponding field name is created in the ingestion pod container at `/etc/velox/external_certs/offload`. The examples here would result in the files `keystore.jks` and `cacert.pem`.

- Save the file.

4. Optional: Create an additional secret.

If you have sensitive data such as a certificate passphrase that must be available in text format, you can store it in another secret.

If your passphrase is not sensitive (such as the JKS default passphrase), then you can supply it directly in the Logstash configuration, and skip this step.

- Create the file `offload_env_var.yaml`. The secret's name that you specify in the file does not have to match the file name. In this example, the secret's name is `offload-jks-passphrase`.
- In the `stringData` section, set the property `jks_password.txt` to your passphrase (in plain text):

```
apiVersion: v1
kind: Secret
metadata:
  name: offload-jks-passphrase
stringData:
  jks_password.txt: <passphrase>
```

- Run the following command to update the cluster:

```
oc apply -f offload_env_var.yaml -n your_namespace
```

Where `offload_env_var.yaml` is the secret's file name.

- Apply the secret to the cluster where the Analytics subsystem is installed. Run the following command to update the cluster:

```
kubectl apply -f file_name_of_secret.yaml -n namespace
```

Where:

- `file_name_of_secret.yaml` is the secret's file name.
- `namespace` is your analytics namespace.

- If using a top-level CR edit your `APIConnectCluster` CR file. Otherwise edit your `analytics_cr.yaml` file and add a reference to the secret and passwordSecret in the `offload` section; for example:

```
offload:
  enabled: true
  output: |
    kafka {
      ...
    }
  secretName: offload-certificates
  passwordSecretName: offload-jks-password
```

7. If you want to reference the certificate in [Offloading data to a third-party system](#), configure the plugin now. Step 3 of that procedure shows how you can include the certificate information in the plugin.

Disabling internal storage

Update the OpenShift top-level or analytics CR to disable local storage in your IBM® API Connect analytics deployment.

About this task

The Analytics subsystem provides a complete solution for routing, storing, and viewing the analytics data. However, you might not need the internal storage and viewing solution, especially if you are primarily using a third-party system for data offloading. In this scenario, you can greatly reduce your CPU and memory costs by disabling the internal storage and viewing components.

With this approach, data that is offloaded is not viewable in the API Manager and Developer Portal UIs.

When you disable internal storage for analytics data, the following microservices are disabled:

- **osinit**
- **storage**

Important: The disablement of internal storage must be set in the analytics CR yml file before installation. It is not possible to disable or re-enable internal storage after installation.

Procedure

1. Required: Configure data offloading as explained in [Offloading data to a third-party system](#). If you do not have offloading configured and also disable internal storage, you will have no analytics data.
2. If using a top-level CR edit your **APIConnectCluster** CR file and find the **spec.analytics** section. Otherwise edit your `analytics_cr.yaml` file and find the **spec** section.
3. Locate the **storage** section and set **enabled** to **false**:

```
storage:
  enabled: false
```

4. Save the file.

Modifying incoming analytics data

Update the OpenShift top-level or analytics CR to customize data in your analytics deployment.

Before you begin

Review the use of filters for modifying analytics data, and make sure that you understand the restrictions. For information about customizing data, see [Planning to modify your incoming analytics data](#).

About this task

You can define filters that add, modify, or remove fields from the incoming API event records, before they are stored in API Connect or offloaded to a third-party system.

Procedure

1. If you are using a top-level CR, edit your **APIConnectCluster** CR file and find the **spec.analytics** section. Otherwise, edit your `analytics_cr.yaml` file and find the **spec** section.
2. Filters for API events that are offloaded to an external system are defined in the **external.offload** section. Filters for API events that are stored in the API Connect OpenSearch database are defined in the **ingestion** section.

Ingestion filters - for API event data stored by API Connect
Add the filter definition to the **ingestion** section:

```
spec:
  ingestion:
    ...
    filter: |
      mutate {
        remove_field => ["query_string"]
      }
}
```


Offload filters - for API event data that is sent to a third-party system.
Add the filter definition to the `external.offload` section:

```
spec:
  external:
    offload:
      filter: |
        mutate {
          remove_field => ["query_string"]
        }
      }
```

For more examples, see [Sample filters for modifying analytics data](#).

3. Optional: Include geoIP fields in your analytics data, for both stored and offloaded data, by inserting `geoIPEnabled: true` in the ingestion section:

```
ingestion:
  ...
  geoIPEnabled: true
```

By default the `client_geoip` and `gateway_geoip` fields are not included in the analytics data. You do not need to add `geoIPEnabled` to the `external.offload` section. Enabling geoIP in the `ingestion` section enables it for both data routes.

Note: To ensure the value in the `client_geoip` field is accurate, the gateway must receive the `X-Forwarded-For` header in all API calls. Check with the administrator of your deployment environment to ensure that the `X-Forwarded-For` header is passed to your gateways. For example, in a Kubernetes environment where NINGX ingresses are used, configure the NGINX ingress that is used by your gateway to use the `X-Forwarded-For` header: [NGINX Configuration "use-forwarded-headers"](#).

4. Save the file. Changes to filters take effect after approximately 30 seconds.

- [Sample filters for modifying analytics](#)

Review sample filters to see how to code your own filters for refining IBM API Connect Analytics data by adding fields, removing fields, or modifying field contents.

Sample filters for modifying analytics

Review sample filters to see how to code your own filters for refining IBM® API Connect Analytics data by adding fields, removing fields, or modifying field contents.

You can use these sample filters in the `ingestion.filter` and the `external.offload.filter` configurations.

Adding a new field

You can add custom data to API event data by adding a field to the logging document. For example, if you want to search or visualize data that is stored in request and response headers, you can add fields to a filter to include that information.

To avoid naming conflicts with future or current analytics fields, include a prefix that ensures your new field name is unique. For example, instead of naming the field `employee_num` you might name it `x_mycompany_employee_num`.

The following example copies the contents of the `X-Employee-Num` field from the request header and adds it to the `x_mycompany_employee_num` field in the event data. Adding the field to the event data enables you to use the information in visualizations.

```
if [request_http_headers] {
  ruby {
    code => "event.get(['request_http_headers']).collect {|i| event.set(['x_mycompany_employee_num'], i['X-Employee-Num']) if
i.has_key?('X-Employee-Num')}"
  }
}
```

Modifying an existing field

Sometimes you don't want to remove a field entirely from your data, you just want to redact sensitive information such as IDs to prevent them from being exposed in visualizations. You can modify the contents of a field and replace information with a symbols or a message.

Remember: The following fields should not be modified if the data is being written to internal analytics storage: `org_id`, `catalog_id`, `space_id`, `developer_org_id`, `datetime`, and `@timestamp`.

The following example replaces sensitive information in the `X-Employee-Name` and `X-Employee-ID` request headers with the string: `*****sanitized*****`.

```
if [request_http_headers] {
  ruby {
    code => "headers=['X-Employee-Name','X-Employee-ID']; newHeaders = event.get(['request_http_headers']).collect {|i|
headers.each {|header| i[header] = '*****sanitized*****' if i.has_key?(header)}; i}; event.set(['request_http_headers'],
newHeaders)"
  }
}
```

Removing an existing field

Use the mutate `remove_field` operation to delete a field from the Analytics data. To remove multiple fields, delimit the field names with commas.

The following example removes multiple fields (`request_http_headers`, `response_http_headers`, `request_body`, `response_body`, and `query_string`) from the Analytics data:

```
mutate {
  remove_field => ["request_http_headers", "response_http_headers", "request_body", "response_body", "query_string"]
}
```

Combining filters with offload and ingestion configuration

Offload events from `catalog1` only:

```
offload:
  enabled: true
  if [catalog_name] == "offload1" {
    http {
      url => "http://offloadhost.example.com/offload"
      http_method => "post"
      codec => "json"
      content_type => "application/json"
      id => "offload_http"
    }
  }
}
```

Offload to different locations based on catalog name:

```
offload:
  enabled: true
  output: |
    if [catalog_name] == "offload1" {
      http {
        url => "http://offloadhost.example.com /offload1"
        http_method => "post"
        codec => "json"
        content_type => "application/json"
        id => "offload1_http"
      }
    }
    else if [catalog_name] == "offload2" {
      http {
        url => "http://offloadhost2.example.com /offload2"
        http_method => "post"
        codec => "json"
        content_type => "application/json"
        id => "offload2_http"
      }
    }
  }
}
```

Use the catalog name as a variable in the offload location:

```
offload:
  enabled: true
  http {
    url => "http://offloadhost.example.com /%{[catalog_name]}"
    http_method => "post"
    codec => "json"
    content_type => "application/json"
    id => "offload_http"
  }
}
```

Do not offload events from `apic` and `sandbox` catalogs:

```
offload:
  enabled: true
  if [catalog_name] !~ /sandbox|apic/ {
    http {
      url => "http://offloadhost.example.com /%{[catalog_name]}"
      http_method => "post"
      codec => "json"
      content_type => "application/json"
      id => "offload_http"
    }
  }
}
```

Do not store events from `apic` and `sandbox` catalogs:

```
ingestion:
  filter: |
    if [catalog_name] =~ /sandbox|apic/ {
      drop { }
    }
}
```

Filter and output plugin for splunk

When you create a filter for use with splunk, you must also create an output plugin for it.

Filter:

```
if "apicapievent" in [tags] {
  ruby {
    code => "event.set('@metadata[newevent]', event.to_json)"
  }
}
```

Output plugin:

```
http {
  url => "http://your-domain/services/collector/event"
  http_method => "post"
  codec => "json"
}
```

```

content_type => "application/json"
id => "offload_http"
format => "message"
message => '{"event": "%{[@metadata] newevent}","index": "applications-int-ms","sourcetype": "json:apimlogs"}'
headers => ["Authorization", "Splunk XXXX"]
}

```

Reference: Fields in the Analytics CR

Use the table of analytics-specific fields to configure the top-level or analytics CR for installing your API Connect analytics subsystem.

If using a top-level CR, make changes there. If you make changes directly to the analytics CR, they will be overwritten by the contents of the top-level CR the next time the API Connect operator reconciliation loop runs.

The two fields that must be in Analytics CR are `ingestion.endpoint` and `ingestion.clientSubjectDN`. The other fields are optional depending on the analytics features you want to configure.

Table 1. Configurable settings in the Analytics subsystem CR.

Field	Type	Default value	Description	Comments
<code>allowUpgrade</code>	Boolean	no	Set to true to allow analytics upgrade to v10.0.5.	See Additional considerations upgrading to analytics v10.0.5 - Kubernetes , Additional considerations upgrading to analytics v10.0.5 - OpenShift .
<code>databaseBackup: host</code>	String		The URL to your S3 storage.	See Configuring backup settings for Analytics , Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration .
<code>databaseBackup: path</code>	String		The path to your S3 storage at the URL specified by <code>databaseBackup.host</code> .	See Configuring backup settings for Analytics , Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration .
<code>databaseBackup: schedule</code>	String		Specifies the frequency of automatic backups. Use any valid <code>cron</code> string.	See Configuring backup settings for Analytics , Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration .
<code>databaseBackup: enableCompression</code>	Boolean	true	Specifies whether to compress the backups.	See Configuring backup settings for Analytics , Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration .
<code>databaseBackup: credentials</code>	String		The name of your S3 storage authentication secret.	See Configuring backup settings for Analytics , Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration .
<code>databaseBackup: chunkSize</code>	String	1Gi		See Configuring backup settings for Analytics , Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration .
<code>databaseBackup: enableServerSideEncryption</code>	Boolean			See Configuring backup settings for Analytics , Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration .
<code>external: offload: filter</code>	String		Filter configuration for external offload.	See Offloading data to a third-party system - Kubernetes , Offloading data to a third-party system - OpenShift .
<code>external: offload: enabled</code>	Boolean	false	Enables analytics offload feature.	See Offloading data to a third-party system - Kubernetes , Offloading data to a third-party system - OpenShift .
<code>external: offload: output</code>	String		Output configuration for external offload.	See Offloading data to a third-party system - Kubernetes , Offloading data to a third-party system - OpenShift .
<code>external: offload: passwordSecretName</code>	String		The name of the secret that has the password to access the certificate keystore.	For where you are using a password protected key/truststore. See Offloading data to a third-party system - Kubernetes , Offloading data to a third-party system - OpenShift .
<code>external: offload: secretName</code>	String		The name of the secret that contains the certificates used for connecting to the offload target.	See Offloading data to a third-party system - Kubernetes , Offloading data to a third-party system - OpenShift .
<code>ingestion: endpoint</code>	Object		FQDN and TLS Secrets for client endpoint.	Identifies the endpoint that enables the Gateway to push your analytics data.
<code>ingestion: clientSubjectDN</code>	String		Subject DN (distinguished name) for authenticating with ingestion endpoint (RFC 2253).	

Field	Type	Default value	Description	Comments
<code>ingestion:filter</code>	String		Filter to modify API event data that is stored internally (not offloaded).	See Configuring ingestion filters on Kubernetes Configuring ingestion filters on OpenShift .
<code>ingestion:queue</code>	Object		Persistent queue settings. Must be set to persisted if present.	See Configuring persistent queue on Kubernetes , Configuring persistent queue on OpenShift .
<code>ingestion:queue:type</code>	String		Defines the storage type for the persistent storage feature. Options are 'persisted' or 'memory'	See Configuring persistent queue on Kubernetes , Configuring persistent queue on OpenShift .
<code>ingestion:queue:volumeClaimTemplate:volumeSize</code>	String		Storage size for your persistent queue.	See Configuring persistent queue on Kubernetes , Configuring persistent queue on OpenShift .
<code>ingestion:queue:volumeClaimTemplate:storageClassName</code>	String		Storage class name for your persistent storage.	See Configuring persistent queue on Kubernetes , Configuring persistent queue on OpenShift .
<code>ingestion:pipelineWorkers</code>	4 <= integer <= 128	24	Number of pipeline workers.	More workers mean more CPU and memory requirements, but gives you the benefit of extra threads to pull data off the ingestion http interface and process it for storage. This has diminishing returns as the size increases, and should usually be multiple of the number of cores on your hardware. Adjust with caution.
<code>ingestion:pipelineBatchSize</code>	10 <= integer <= 10000	250	Batch size of a pipeline worker.	Pulling more or less data off the http interface per worker can have benefits. Recommended values are 125, 250, and 500. Increasing the value requires more memory.
<code>ingestion:pipelineBatchDelay</code>	10 <= integer <= 5000	50	Batch delay of a pipeline worker.	How long to wait for the batch size to fill before leaving the http interface to process for storage. Having threads wait too long causes idle threads in your system; however, setting this value too low decreases performance. Adjust with caution.
<code>ingestion:configReloadAutomatic</code>	Boolean	true	Whether to automatically reload the pipeline on configuration changes.	Typically, when the underlying files for the pipeline are modified, the pipeline process automatically reloads and picks up the changes. However, in Kubernetes environments where the pipeline is mapped to configmaps, this is unreliable. Ingestion microservice restarts are usually required.
<code>ingestion:pipelineUnsafeShutdown</code>	Boolean	false	Whether to allow unsafe shutdown of the pipeline.	Typically, the pipeline attempts to empty all workers' batches before shutdown to prevent data loss. You might need to enable this setting if the pipeline is badly configured and refuses to respond to shut down requests (uncommon). Adjust with caution.
<code>standaloneCluster</code>	Boolean			
<code>storage:type</code>	String	shared	Type of storage deployment.	Shared is the only available option.
<code>storage:shared:volumeClaimTemplate:storageClassName</code>	String	cluster default	Storage class name.	Name of the storage class configured to use for creating the microservice persistent volume.
<code>storage:shared:volumeClaimTemplate:volumeSize</code>	String	50Gi	Volume size.	Size of the volume used for data storage. See Estimating internal storage space for more information.

Additional Kubernetes settings

For each API Connect subsystem, you can specify additional settings for annotations and pod management.

The additional settings are generic Kubernetes configuration settings. The Platform UI advanced configuration option for API Connect enables you to set them specifically for each subsystem.

Annotations

The annotations field serves as a pass-through for all the resources managed by the CR. You can add any annotation to this field. The annotations here overwrite the default annotations.

See <https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations/>.

Labels

The labels field serves as a pass-through for Pod labels. You can add any label to this field and have it apply to the pod. The labels here overwrite the default labels provided. For example, if the label 'mylabel=1' was provided, it would apply to all kubernetes resources managed by this CR.

See <https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/>.

Node Selector

This field allows you to specify a set of key value pair that must be matched against the node labels to decide which API Connect pods can be scheduled on that node. Only nodes matching all of these key value pairs in their labels will be selected for scheduling API Connect pods.

See <https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/>.

Tolerations

This field allows you to select which nodes API Connect pods can be scheduled or not scheduled. By specifying a set of matching tolerations, you can control whether specific nodes can be used for scheduling API Connect Pods.

Effect (optional)

Effect indicates the taint effect to match. Empty means match all taint effects. When specified, allowed values are NoSchedule, PreferNoSchedule and NoExecute.

Key (optional)

Key is the taint key that the toleration applies to. Empty means match all taint keys. If the key is empty, operator must be Exists; this combination means to match all values and all keys.

Operator represents a key's relationship to the value. Valid operators are Exists and Equal. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a particular category.

Operator (optional)

Operator represents a key's relationship to the value. Valid operators are Exists and Equal. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a particular category.

Toleration seconds (optional)

TolerationSeconds represents the period of time the toleration (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By default, it is not set, which means tolerate the taint forever (do not evict). Zero and negative values will be treated as 0 (evict immediately) by the system.

Value (optional)

Value is the taint value the toleration matches to. If the operator is Exists, the value should be empty, otherwise just a regular string.

See <https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/>.

Configuring timeouts for management endpoints

Adjust the timeout settings for management endpoints by updating the CR.

About this task

On large deployments, or with large API usage, you might encounter timeout issues when deployed with the default timeout setting of 30 seconds. To avoid timeout issues, increase the setting for the management endpoints used in routing.

You do not have to set these values on initial installation. You can set them or change them after installation and reapply the CR YAML files.

Procedure

Edit the CR to set the timeout to 240 seconds.

The instructions for editing depend on whether you install API Connect using subsystem CRs or a single top-level CR:

- If you are installing API Connect using subsystem CRs, edit `management_CR.yaml`:

```
cloudManagerEndpoint:
  annotations:
    haproxy.router.openshift.io/timeout: 240s
  hosts:
  - name: admin.example.com
    secretName: cm-endpoint
apiManagerEndpoint:
  annotations:
    haproxy.router.openshift.io/timeout: 240s
  hosts:
  - name: manager.example.com
    secretName: apim-endpoint
platformAPIEndpoint:
  annotations:
    haproxy.router.openshift.io/timeout: 240s
  hosts:
  - name: api.example.com
    secretName: api-endpoint
consumerAPIEndpoint:
  annotations:
```

```
haproxy.router.openshift.io/timeout: 240s
hosts:
- name: consumer.example.com
  secretName: consumer-endpoint
```

- If you are installing API Connect using a top-level CR, the management subsystem settings are embedded in the top-level APICoconnectCluster CR. Edit the endpoint settings under `spec:management:` as follows:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APICoconnectCluster
metadata:
  labels:
    app.kubernetes.io/instance: apiconnect
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: apiconnect-minimum
  name: <name_of_your_instance>
  namespace: ${NAMESPACE}
spec:
  ...
  management:
    cloudManagerEndpoint:
      annotations:
        haproxy.router.openshift.io/timeout: 240s
      hosts:
        - name: admin.example.com
          secretName: cm-endpoint
    apiManagerEndpoint:
      annotations:
        haproxy.router.openshift.io/timeout: 240s
      hosts:
        - name: manager.example.com
          secretName: apim-endpoint
    platformAPIEndpoint:
      annotations:
        haproxy.router.openshift.io/timeout: 240s
      hosts:
        - name: api.example.com
          secretName: api-endpoint
    consumerAPIEndpoint:
      annotations:
        haproxy.router.openshift.io/timeout: 240s
      hosts:
        - name: consumer.example.com
          secretName: consumer-endpoint
  ...
```

Installation procedures with IBM Cloud Pak for Integration

The API Management capability of IBM Cloud Pak for Integration uses IBM API Connect.

There are several methods for installing API Connect with Cloud Pak for Integration:

- Use the Cloud Pak for Integration Platform UI to install API Connect with a top-level CR. The steps are documented in the installation section of the Cloud Pak for Integration documentation: <https://www.ibm.com/docs/en/cloud-paks/cp-integration>.
- If API Connect is already installed with the top-level CR and you want to migrate it to Cloud Pak for Integration, see [Migrating from OpenShift to Cloud Pak for Integration on the same cluster](#).

Installing with the top-level CR on OpenShift

Use the single, top-level `APICoconnectCluster` CR to deploy API Connect on OpenShift.

About this task

When you deploy API Connect with a top-level CR, all of the subsystems are deployed in the same namespace. The CR contains a section for each subsystem; configure deployment settings in the appropriate section for each subsystem.

If you want to install multiple API Connect instances, for example dev, uat, and prod, the recommended configuration is to have single API Connect and Datapower operators installed globally in the `openshift-operators` namespace and each API Connect instance installed in its own namespace using a top-level CR.

- [Connected installation](#)
Install API Connect on OpenShift using a single, top-level Custom Resource while connected to the internet.
- [Air-gapped installation](#)
In a disconnected environment using Red Hat OpenShift Container Platform (OCP), install API Connect by mirroring product images to a bastion host or a portable device and then completing the installation.
- [Enable additional features post-install](#)
After installing API Connect, you can enable additional features that are related to inter-subsystem communication.

Connected installation

Install API Connect on OpenShift using a single, top-level Custom Resource while connected to the internet.

About this task

Complete the following tasks in the sequence shown:

1. [Installing operators](#)
Install operators for API Connect, DataPower, and common services so that you can deploy API Connect.
2. [Installing API Connect](#)
Use the top-level `APIConnectCluster` CR to install all of the API Connect subsystems into a shared namespace on OpenShift while connected to the internet.

Installing operators

Install operators for API Connect, DataPower, and common services so that you can deploy API Connect.

Before you begin

The API Connect operator and operand must be from the same release and fix pack level. Table 1 lists the current version of the operator and operand for API Connect.

Table 1. API Connect operator and operand versions

API Connect (operand)	Operator channel version	Highest operator version
10.0.5.4	v3.4	3.4.0

You can use any combination of API Connect 10.0.5.x with DataPower Gateway 10.5.0.x or DataPower API Gateway 10.5.0.x.

Procedure

1. Obtain an entitlement key for the Entitled Registry:
 - a. Log in to the [IBM Container Library](#).
 - b. In the Container software library, select Get entitlement key.
 - c. After the Access your container software heading, click Copy key.
 - d. Copy the key to a safe location.
2. Create the namespace where API Connect will be installed, either by selecting Home->Projects->Create Project in the UI, or with the command:

```
oc create ns <APIC-namespace>
```

The namespace where you install API Connect must meet the following requirements:

- OpenShift restricts the use of default namespaces for installing non-cluster services. The following namespaces cannot be used to install API Connect: `default`, `kube-system`, `kube-public`, `openshift-node`, `openshift-infra`, `openshift`.
- OpenShift: Only one top-level CR (APIConnectCluster) can be deployed in each namespace.
- Cloud Pak for Integration: Only one API Connect capability can be deployed in each namespace.

3. If you are installing the operator into a single namespace, create an OperatorGroup object specifying that namespace.

Attention: If you are installing the operator into all namespaces, skip this step because the `openshift-operators` namespace already has an appropriate Operator group in place.

- a. Create the `apiconnect-operator-group.yaml` file with an OperatorGroup object that identifies the namespace in which RBAC permissions for the IBM API Connect Operator will be generated, and where the CSV will be available:

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: <operator-group-name>
  namespace: <APIC-namespace>
spec:
  targetNamespaces:
  - <APIC-namespace>
```

- b. Apply the OperatorGroup object with the following command:

```
oc apply -f apiconnect-operator-group.yaml
```

4. Create the image pull secret in the namespace where you want to install API Connect:
 - a. Open the OpenShift web console, click Workloads > Secrets.
 - b. Ensure that the Project is set to the namespace where you intend to install API Connect.
 - c. Click Create and select Image pull secret.
 - d. Set the following parameters for the secret:
 - Set Secret name to `ibm-entitlement-key`.
 - Set Authentication type to `Image registry credentials`.
 - Set Registry server address to `cp.icr.io`.
 - Set Username to `cp`.
 - Set Password to the entitlement key generated in Step 1.
 - Click Create to create the secret.

Note: If your OpenShift platform is Red Hat OpenShift on IBM Cloud (ROKS), you might need to reload your worker nodes after the image pull secret is created. You can reload the worker nodes either from the OpenShift web console or from the command line with: `oc worker reload`.

5. Add the IBM Operator Catalog source:
 - a. In the OpenShift web console, and click + (in the page banner) to open the Import YAML dialog box.
 - b. Paste the following resource definition into the dialog box:

```

apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: ibm-operator-catalog
  namespace: openshift-marketplace
spec:
  displayName: "IBM Operator Catalog"
  publisher: IBM
  sourceType: grpc
  image: icr.io/cpopen/ibm-operator-catalog
  updateStrategy:
    registryPoll:
      interval: 45m

```

c. Click Create.

6. Create the `apiconnect` subscription.

a. Create the `apiconnect` subscription with the appropriate channel by creating a file called `apic-sub.yaml` and pasting in the following contents, updating the namespace as required:

```

apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-apiconnect
  namespace: <namespace>
spec:
  channel: v3.4
  name: ibm-apiconnect
  source: ibm-operator-catalog
  sourceNamespace: openshift-marketplace

```

Where:

- `<namespace>` is set to `openshift-operators` if you are installing the operator in all namespaces. Otherwise set this to the namespace created in [step 2](#).

b. Apply the subscription with the following command:

```
oc apply -f apic-sub.yaml
```

c. Select Operators, Installed Operators, and ensure that Project: All Projects is selected.

If any operators such as `ibm-apiconnect` or `ibm-cert-manager-operator` show the status of "Upgrade available", approve the upgrade by completing the following steps:

- Click Upgrade available.
- Click Preview InstallPlan.
- Click Approve.
- Wait for the `IBM API Connect` and `IBM DataPower Gateway` operators to install.

The `IBM DataPower Gateway` operator is a prerequisite to API Connect and must not be removed.

7. (10.0.5.3 and later) Create the `ibm-common-services-operator` subscription, if it does not already exist.

This step only applies when you install version 10.0.5.3 or later. If you are installing an older version of API Connect, skip this step.

a. Create a file called `common-services-sub.yaml` and paste in the following contents:

```

apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-common-service-operator
  namespace: <namespace>
spec:
  channel: <channel>
  name: ibm-common-service-operator
  source: ibm-operator-catalog
  sourceNamespace: openshift-marketplace

```

Where:

- `<namespace>` is one of the following values:
 - `openshift-operators` if you are installing the operator in all namespaces
 - The namespace created in [step 2](#) if you are installing the operator in a single namespace
- `<channel>` is one of the following values:
 - `v3.23` if you are using IBM Cloud Pak foundational services for CD (continuous delivery)
 - `v3` if you are using IBM Cloud Pak foundational services for LTSR (long term service release)

b. Apply the subscription with the following command:

```
oc apply -f common-services-sub.yaml
```

c. Select Operators, Installed Operators, and ensure that Project: All Projects is selected.

If any operators such as `ibm-apiconnect` or `ibm-cert-manager-operator` show the status of "Upgrade available", approve the upgrade by completing the following steps:

- Click Upgrade available.
- Click Preview InstallPlan.
- Click Approve.
- Check the `ibm-common-services` namespace and ensure that all operators with a status of "Upgrade available" are approved.
- Wait for the `IBM Cloud Pak foundational services`, `IBM NamespaceScope`, and `Operand Deployment Lifecycle Manager` operators to install.

8. (10.0.5.3 and later) Install cert manager:

This step only applies when you install version 10.0.5.3 or later. If you are installing an older version of API Connect, skip this step.

a. Create a file called `cert-manager-operand-request.yaml` and paste in the following content:


```

apiVersion: operator.ibm.com/v1alpha1
kind: OperandRequest
metadata:
  name: ibm-apiconnect-cert-manager
  namespace: <namespace>
spec:
  requests:
  - operands:
    - name: ibm-cert-manager-operator
      registry: common-service
      registryNamespace: ibm-common-services

```

b. Create the `operandRequest` for cert-manager by running the following command:

```
oc apply -f cert-manager-operand-request.yaml
```

c. Select Operators, Installed Operators, and ensure that Project: All Projects is selected.

Wait for the `ibm-cert-manager-operator` to display and if it shows the status of "Upgrade available", approve the upgrade by completing the following steps:

- i. Click Upgrade available.
- ii. Click Preview InstallPlan.
- iii. Click Approve.
- iv. Wait for the **IBM Cert Manager** operator to install.

Next topic: [Installing API Connect](#)

Installing API Connect

Use the top-level `APICoconnectCluster` CR to install all of the API Connect subsystems into a shared namespace on OpenShift while connected to the internet.

Before you begin

Review installation considerations and complete and prerequisite tasks as explained in:

1. [Preparing for installation](#)
2. [Installing operators](#)

Procedure

1. Install the subsystems by using the OpenShift web console.

API Connect uses a single top-level `apiconnectcluster` CR to quickly deploy all of the subsystems. With the top-level CR, you provide values for a small number of properties, and then the install logic chooses sensible defaults for you while deploying all of the subsystems.

- a. Select Operators->Installed Operators
- b. Click on the IBM API Connect operator.
- c. Click Create Instance on the API Connect cluster tile to install the top level Custom Resource.

Selecting API Connect cluster installs all the API Connect subsystems with the default configuration. The API Connect subsystems are Management, Developer Portal, Gateway, and Analytics.

d. Provide values for the following fields.

Table 1. Properties for the API Connect top-level CR

Input	Value
Name	The name to be used when deploying the API Connect cluster Custom Resource
License acceptance	Required. Click to accept the license. You must accept the license to install API Connect.
License use	Required. Select production or nonproduction to match the type of license that you purchased.
License metric	Optional. Enter the unit of measurement that is used for your program license: <ul style="list-style-type: none"> • PROCESSOR_VALUE_UNIT - Default value. If you leave the field blank, this value is used. • MONTHLY_API_CALL - Applies only to the IBM API Connect Hybrid Entitlement program.
License ID	Required. Select the license ID for the API Connect program that you purchased. To view all license IDs, see API Connect licenses .
Deployment profile	The deployment profile defines the number of worker nodes, CPU, and memory that is available for API Connect. The available profiles are listed in API Connect deployment profiles for OpenShift and Cloud Pak for Integration .
Product version	Optional. The API Connect application version. Defaults to the latest version available.
Default storage class name	The name of the Storage class to be used You can set a default to appear here with the command: <pre>oc patch storageclass <storage class name> -p '{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-class": "true"}}}'</pre>

e. If you want to configure additional settings, click Advanced and see [API Connect configuration settings](#) for information on the advanced settings.

Notes:

- Before you install API Connect, you can customize the configuration of the analytics subsystem. Review the information in [Planning your analytics deployment](#) to choose configuration options, and then configure the settings as explained in [Analytics subsystem CR configured settings](#). The OpenShift web console deploys API Connect with a single, top-level CR, so add your configuration settings to the `spec.analytics` section of the CR.
- When setting additional values, you cannot specify more than one host for the `portalUIEndpoint` endpoint, due to a limitation of the Routes in OCP. If you require multiple `portalUIEndpoint` hosts to provide public access to multiple catalogs, deploy a new Portal service for each host name.

2. To verify your API Connect cluster is successfully installed, run `oc get apic -n <APIC-namespace>`.

- Verify that you can log in to the API Connect Cloud Manager UI:
 - To determine the location for logging in, view all the endpoints:

```
oc get routes -n <APIC-namespace>
```

- Locate the `mgmt-admin-apic` endpoint, and access the Cloud Manager UI
- Login as `admin`. When you install with the top-level CR, the password is auto-generated. To get the password:

```
oc get secret -n <APIC-namespace> | grep mgmt-admin-pass
```

```
oc get secret -n <APIC-namespace> <secret_name_from_previous_command> -o jsonpath="{.data.password}" | base64 -d && echo ""
```

- Optional: Configure additional features related to inter-subsystem communication security, such as CA verification and JWT security: [Enable additional features post-install.](#)
- Optional: Increase the timeout settings for management endpoints, particularly for large deployments. See [Configuring timeouts for management endpoints.](#)
- Optional tasks:
 - Toolkit**
You can optionally download the API Connect toolkit from the Cloud Manager UI or API Manager UI after installation is completed. See [Installing the toolkit.](#)
 - Local Test Environment**
You can install this optional test environment after initial installation. See [Testing an API with the Local Test Environment.](#)

Note: You can also install the Toolkit and Local Test Environment from IBM Fix Central. For a link to the latest files on Fix Central, see [What's new in the latest release.](#)

What to do next

When you finish installing API Connect, [prepare your deployment for disaster recovery](#) so that your data can be restored in the event of an emergency.

Previous topic: [Installing operators](#)

Air-gapped installation

In a disconnected environment using Red Hat OpenShift Container Platform (OCP), install API Connect by mirroring product images to a bastion host or a portable device and then completing the installation.

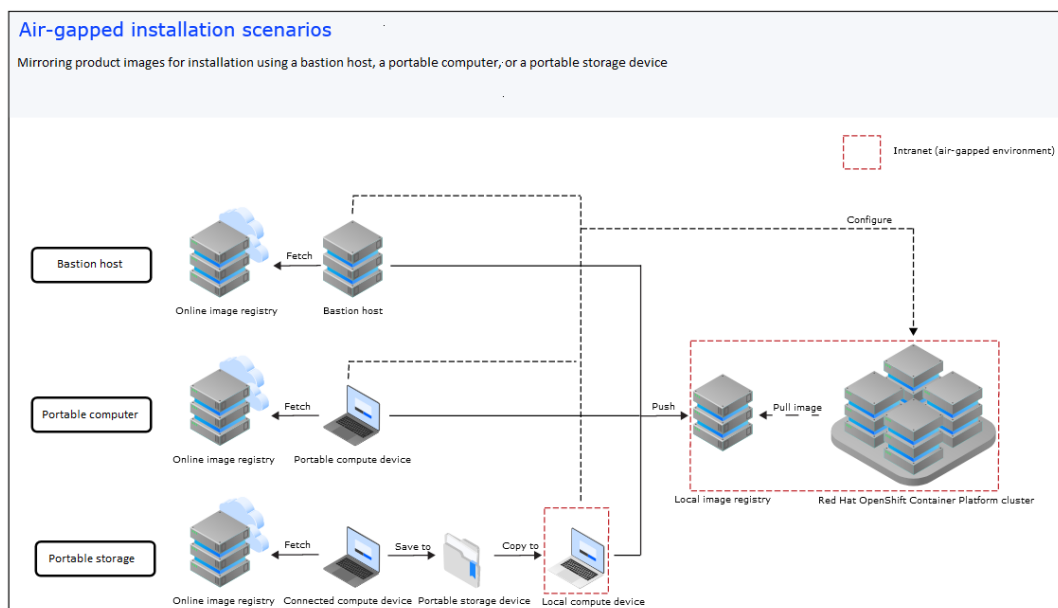
Before you begin

- Mirroring API Connect product images requires the use of IBM Catalog Management Plug-in for IBM Cloud Paks because IBM Cloud Pak CLI (cloudctl) is deprecated. If you still require the use of cloudctl for mirroring product images, see [Deprecated: Air-gapped installation using cloudctl.](#)
- An air-gapped installation is not possible in the following environments:
 - Red Hat OpenShift Service on AWS (ROSA)
 - Red Hat OpenShift Service Kubernetes Service on IBM Cloud (ROKS)

About this task

If your cluster is not connected to the internet (it is network-restricted, or air-gapped), you can install API Connect in your cluster by using either a bastion host, a portable computer, or a portable storage device.

The following diagram illustrates how you can mirror product images for an air-gapped installation with a bastion host, a portable computer, or a portable storage device.



All of the scenarios use Container Application Software for Enterprises (CASE) files to mirror content from a source (on the internet) to an intermediary target (a bastion host, a portable computer, or a portable storage device) before pushing them to a local server or device.

Then, you can transfer the files to the local air-gapped network and run the installation.

- **Installing with a bastion host**
You can use a bastion host to perform an air-gapped installation of API Connect on Red Hat OpenShift Container Platform (OCP) when your cluster has no internet connectivity.
- **Installing with a portable computer or storage device**
You can use a portable device, such as a portal computer or a USB storage device, to perform an air-gapped installation of API Connect on Red Hat OpenShift Container Platform (OCP) when your cluster has no internet connectivity.

Installing with a bastion host

You can use a bastion host to perform an air-gapped installation of API Connect on Red Hat OpenShift Container Platform (OCP) when your cluster has no internet connectivity.

Before you begin

This task must be performed by a Red Hat OpenShift administrator.

About this task

If your cluster is not connected to the internet, you can mirror product images to a registry in your network-restricted environment by using a bastion host. A bastion host has access to both the public internet and the network-restricted environment where the target clusters reside. You can fetch product images from the internet and push them to a local registry; then you can pull the images from the local registry to the target cluster for installation.

Note: In 10.0.5.3 and later, the API Connect operator is uncoupled from the IBM Cloud Pak foundational services operator. To accommodate the change, you will create environment variables and run commands to download and mirror two CASE files instead of one CASE file as in previous releases.

Procedure

1. Set up the mirroring environment.
 - a. Prepare the target cluster:
 - Deploy a supported version of Red Hat OpenShift Container Platform (OCP) as a cluster.
For information, see Table 2 "API Connect and OpenShift Container Platform (OCP) compatibility matrix" in [IBM API Connect Version 10 software product compatibility requirements](#).
 - Configure storage on the cluster and make sure that it is available.
 - b. Prepare the bastion host:
You must be able to connect your bastion host to the internet and to the restricted network environment (with access to the Red Hat OpenShift Container Platform (OCP) cluster and the local registry) at the same time. Your host must be on a Linux x86_64 or Mac platform with any operating system that the Red Hat OpenShift Client supports (in Windows, execute the actions in a Linux x86_64 VM or from a Windows Subsystem for Linux terminal).

- i. Ensure that the sites and ports listed in Table 1 can be reached from the bastion host:

Table 1. Sites that must be reached from the bastion host

Site	Description
icr.io:443	IBM entitled registry
quay.io:443	Local API Connect image repository
github.com	CASE files and tools
redhat.com	Red Hat OpenShift Container Platform (OCP) upgrades

- ii. On the bastion host, install either Docker or Podman (not both).
Docker and Podman are used for managing containers; you only need to install one of these applications.
 - To install Docker (for example, on Red Hat Enterprise Linux), run the following commands:

```
yum check-update
yum install docker
```
 - To install Podman, see the [Podman installation instructions](#).
For example, on Red Hat Enterprise Linux 9, install Podman with the following command:

```
yum install podman
```
- iii. Install the Red Hat OpenShift Client tool (`oc`) as explained in [Getting started with the OpenShift CLI](#).
The `oc` tool is used for managing Red Hat OpenShift resources in the cluster.
- iv. Download the IBM Catalog Management Plug-in for IBM Cloud Paks version 1.1.0 or later from [GitHub](#).
The `ibm-pak` plug-in enables you to access hosted product images, and to run `oc ibm-pak` commands against the cluster. To confirm that `ibm-pak` is installed, run the following command and verify that the response lists the command usage:

```
oc ibm-pak --help
```

- c. Set up a local image registry and credentials.
The local Docker registry stores the mirrored images in your network-restricted environment.

- i. Install a registry, or get access to an existing registry.

You might already have access to one or more centralized, corporate registry servers to store the API Connect images. If not, then you must install and configure a production-grade registry before proceeding.

The registry product that you use must meet the following requirements:

- Supports multi-architecture images through Docker Manifest V2, Schema 2
For details, see [Docker Manifest V2, Schema 2](#).
- Is accessible from the Red Hat OpenShift Container Platform cluster nodes
- Allows path separators in the image name

Note: Do not use the Red Hat OpenShift image registry as your local registry because it does not support multi-architecture images or path separators in the image name.

ii. Configure the registry to meet the following requirements:

- Supports auto-repository creation
- Has sufficient storage to hold all of the software that is to be transferred
- Has the credentials of a user who can create and write to repositories (the mirroring process uses these credentials)
- Has the credentials of a user who can read all repositories (the Red Hat OpenShift Container Platform cluster uses these credentials)

To access your registries during an air-gapped installation, use an account that can write to the target local registry. To access your registries during runtime, use an account that can read from the target local registry.

2. Set environment variables and download CASE files.

a. Create the following environment variables with the installer image name and the image inventory on your host:

```
export CASE_NAME=ibm-apiconnect
export CASE_VERSION=4.0.5
export ARCH=amd64
```

For information on API Connect CASE versions and their corresponding operators and operands, see [Operator, operand, and CASE versions](#).

b. Connect your host to the internet (it does not need to be connected to the network-restricted environment at this time).

c. Download the CASE file to your host:

```
oc ibm-pak get $CASE_NAME --version $CASE_VERSION
```

If you omit the `--version` parameter, the command downloads the latest version.

3. Mirror the images.

The process of mirroring images pulls the images from the internet and pushes them to your local registry. After mirroring your images, you can configure your cluster and pull the images to it before installing API Connect.

a. Generate mirror manifests.

i. Define the environment variable `$TARGET_REGISTRY` by running the following command:

```
export TARGET_REGISTRY=<target-registry>
```

Replace `<target-registry>` with the IP address (or host name) and port of the local registry; for example: `172.16.0.10:5000`. If you want the images to use a specific namespace within the target registry, you can specify it here; for example: `172.16.0.10:5000/registry_ns`.

ii. Generate mirror manifests by running the following command:

```
oc ibm-pak generate mirror-manifests $CASE_NAME --version $CASE_VERSION $TARGET_REGISTRY
```

If you need to filter for a specific image group, add the parameter `--filter <image_group>` to the command.

The `generate` command creates the following files at `~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION`:

- `catalog-sources.yaml`
- `catalog-sources-linux-<arch>.yaml` (if there are architecture-specific catalog sources)
- `image-content-source-policy.yaml`
- `images-mapping.txt`

The files are used when mirroring the images to the `TARGET_REGISTRY`.

b. Obtain an entitlement key for the entitled registry where the images are hosted:

- Log in to the [IBM Container Library](#).
- In the Container software library, select Get entitlement key.
- In the "Access your container software" section, click Copy key.
- Copy the key to a safe location; you will use it to log in to `cp.icr.io` in the next step.

c. Authenticate with the entitled registry where the images are hosted.

The image pull secret allows you to authenticate with the entitled registry and access product images.

i. Run the following command to export the path to the file that will store the authentication credentials that are generated on a Podman or Docker login:

```
export REGISTRY_AUTH_FILE=$HOME/.docker/config.json
```

The authentication file is typically located at `$HOME/.docker/config.json` on Linux or `%USERPROFILE%\.docker/config.json` on Windows.

ii. Log in to the `cp.icr.io` registry with Podman or Docker; for example:

```
podman login cp.icr.io
```

Use `cp` as the username and your entitlement key as the password.

d. Authenticate with the local registry.

Log in to the local registry using an account that can write images to that registry; for example:

```
podman login $TARGET_REGISTRY
```

If the registry is insecure, add the following flag to the command: `--tls-verify=false`.

e. Update the CASE manifest to correctly reference the DataPower Operator image.

Files for the DataPower Operator are now hosted on `icr.io`; however, the CASE manifest still refers to `docker.io` as the image host. To work around this issue, visit [Airgap install failure due to 'unable to retrieve source image docker.io'](#) in the DataPower documentation and update the manifest as instructed. After the manifest is updated, continue to the next step in this procedure.

- f. Mirror the product images.
 - i. Connect the bastion host to both the internet and the restricted-network environment that contains the local registry.
 - ii. Run the following command to copy the images to the local registry:

```
oc image mirror \  
-f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping.txt \  
-a $REGISTRY_AUTH_FILE \  
--filter-by-os '*' \  
--skip-multiple-scopes \  
--max-per-registry=1
```

Note: If the local registry is not secured by TLS, or the certificate presented by the local registry is not trusted by your device, add the `--insecure` option to the command.

There might be a slight delay before you see a response to the command.

- g. Configure the target cluster.

Now that images have been mirrored to the local registry, the target cluster must be configured to pull the images from it. Complete the following steps to configure the cluster's global pull secret with the local registry's credentials and then instruct the cluster to pull the images from the local registry.

- i. Log in to your Red Hat OpenShift Container Platform cluster:

```
oc login <openshift_url> -u <username> -p <password> -n <namespace>
```

- ii. [Update the global image pull secret](#) for the cluster as explained in the Red Hat OpenShift Container Platform documentation.

Note: If you have an insecure registry, add the registry to the cluster's `insecureRegistries` list by running the following command:

```
oc edit image.config.openshift.io/cluster -o yaml
```

and add the `TARGET_REGISTRY` to `spec.registrySources.insecureRegistries` as shown in the following example:

```
spec:  
  registrySources:  
    insecureRegistries:  
      - insecure0.svc:5001  
      - <TARGET_REGISTRY>
```

If the `insecureRegistries` field does not exist, you can add it.

- iii. Create the `ImageContentSourcePolicy`, which instructs the cluster to pull the images from your local registry:

```
oc apply -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/image-content-source-policy.yaml
```

- iv. Verify that the `ImageContentSourcePolicy` resource was created:

```
oc get imageContentSourcePolicy
```

- v. Verify your cluster node status:

```
oc get MachineConfigPool -w
```

Wait for all nodes to be updated before proceeding to the next step.

4. Apply the catalog sources.

Now that you have mirrored images to the target cluster, apply the catalog sources.

In the following steps, replace `<Architecture>` with either `amd64`, `s390x` or `ppc64le` as appropriate for your environment.

- a. Export the variables for the command line to use:

```
export CASE_NAME=ibm-apiconnect  
export CASE_VERSION=4.0.5  
export ARCH=amd64
```

- b. Generate the catalog sources and save them in another directory in case you need to replicate this installation in the future.

- i. Get the catalog sources:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
```

- ii. Get any architecture-specific catalog sources that you need to back up as well:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

You can also navigate to the directory in your file browser to copy these artifacts into files that you can keep for re-use or for pipelines.

- c. Apply the catalog sources to the cluster.

- i. Apply the universal catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
```

- ii. Apply any architecture-specific catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

- iii. Confirm that the catalog sources were created in the `openshift-marketplace` namespace:

```
oc get catalogsource -n openshift-marketplace
```

5. Create the namespace where you will install API Connect.

- a. Specify the namespace where you want to install the operator:

```
export NAMESPACE=<APIC-namespace>
```

The namespace where you install API Connect must meet the following requirements:

- Red Hat OpenShift Container Platform (OCP): Only one top-level CR (APIConnectCluster) can be deployed in each namespace.
- Cloud Pak for Integration: Only one API Connect capability can be deployed in each namespace.
- The following namespaces cannot be used to install API Connect because Red Hat OpenShift Container Platform (OCP) restricts the use of default namespaces for installing non-cluster services:
 - `default`
 - `kube-system`
 - `kube-public`
 - `openshift-node`
 - `openshift-infra`
 - `openshift`

b. Create a new namespace for installing the operator:

```
oc new-project $NAMESPACE
```

c. Create an `OperatorGroup`:

i. Create a YAML file called `apiconnect-operator-group.yaml` similar to the following example, replacing `<APIC-namespace>` with your new namespace:

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: ibm-apiconnect-operatorgroup
spec:
  targetNamespaces:
  - <APIC-namespace>
```

ii. Add the new operator group to your namespace:

```
oc apply -f apiconnect-operator-group.yaml -n ${NAMESPACE}
```

6. (10.0.5.3 and later) Create the `ibm-common-services-operator` subscription, if it does not already exist.

This step only applies when you install version 10.0.5.3 or later. If you are installing an older version of API Connect, skip this step.

The `ibm-common-services-operator` is provided by IBM Cloud Pak foundational services.

a. Create a file called `common-services-sub.yaml` and paste in the following contents:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-common-service-operator
  namespace: <APIC-namespace>
spec:
  channel: <channel>
  installPlanApproval: Automatic
  name: ibm-common-service-operator
  source: opencloud-operators
  sourceNamespace: openshift-marketplace
```

Where:

- `<APIC-namespace>` the namespace where you will install API Connect.
- `<channel>` is one of the following values:
 - `v3.23` if you are using IBM Cloud Pak foundational services for Continuous Delivery
 - `v3` if you are using IBM Cloud Pak foundational services for Long Term Service Release

b. Apply the subscription with the following command:

```
oc apply -f common-services-sub.yaml
```

c. Select `Operators`, `Installed Operators`, and ensure that `Project: All Projects` is selected.

If any operators such as `ibm-apiconnect` or `ibm-cert-manager-operator` show the status of "Upgrade available", approve the upgrade by completing the following steps:

- Click Upgrade available.
- Click Preview InstallPlan.
- Click Approve.
- Check the `ibm-common-services` namespace and ensure that all operators with a status of "Upgrade available" are approved.
- Wait for the `IBM Cloud Pak foundational services, IBM NamespaceScope`, and `Operand Deployment Lifecycle Manager` operators to install.

7. (10.0.5.3 and later) Install cert manager:

This step only applies when you install version 10.0.5.3 or later. If you are installing an older version of API Connect, skip this step.

a. Create a file called `cert-manager-operand-request.yaml` and paste in the following content:

```
apiVersion: operator.ibm.com/v1alpha1
kind: OperandRequest
metadata:
  name: ibm-apiconnect-cert-manager
  namespace: <APIC-namespace>
spec:
  requests:
  - operands:
    - name: ibm-cert-manager-operator
      registry: common-service
      registryNamespace: ibm-common-services
```

b. Create the `operandRequest` for cert-manager by running the following command:

```
oc apply -f cert-manager-operand-request.yaml
```

c. Select Operators, Installed Operators, and ensure that Project: All Projects is selected.

Wait for the `ibm-cert-manager-operator` to display and if it shows the status of "Upgrade available", approve the upgrade by completing the following steps:

- i. Click Upgrade available.
- ii. Click Preview InstallPlan.
- iii. Click Approve.
- iv. Wait for the **IBM Cert Manager** operator to install.

8. Create a **Subscription** for the **IBM**

APIConnect operator.

a. Create a YAML file called `apic-sub.yaml` similar to the following example:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-apiconnect
spec:
  channel: v3.4
  name: ibm-apiconnect
  source: ibm-apiconnect-catalog
  sourceNamespace: openshift-marketplace
```

b. Apply the new subscription to your namespace:

```
oc apply -f apic-sub.yaml -n ${NAMESPACE}
```

9. Install API Connect (the operand).

API Connect provides one top-level CR that includes all of the API Connect subsystems (Management, Developer Portal, Analytics, and Gateway).

Tip: Before you install API Connect, you can customize the configuration of the analytics subsystem. Review the information in [Planning your analytics deployment](#) to choose configuration options, and then configure the settings as explained in [Analytics subsystem CR configured settings](#). If you are installing with the top-level **APIConnectCluster** CR, add your configuration settings to the `spec.analytics` section of the CR; if you are installing the subsystems individually, make your changes to the **AnalyticsCluster** CR.

a. Create a YAML file to use for deploying the top-level **APIConnectCluster** CR. Use the template that applies to your deployment (non-production or production).

Note: The values shown in the following examples might not be suitable for your deployment. For information on the license, profile, and version settings, as well as additional configuration settings, see [API Connect configuration settings](#).

- Example CR settings for a one replica deployment:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APIConnectCluster
metadata:
  labels:
    app.kubernetes.io/instance: apiconnect
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: apiconnect-minimum
  name: <name_of_your_instance>
  namespace: <APIC-namespace>
spec:
  license:
    accept: true
    license: L-GVEN-GFUPVE
    metric: PROCESSOR_VALUE_UNIT
    use: nonproduction
  profile: n1xc17.m48
  version: 10.0.5.4
  storageClassName: <default-storage-class>
```

- Example CR settings for a three replica deployment:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APIConnectCluster
metadata:
  labels:
    app.kubernetes.io/instance: apiconnect
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: apiconnect-production
  name: <name_of_your_instance>
  namespace: <APIC-namespace>
spec:
  license:
    accept: true
    license: L-GVEN-GFUPVE
    metric: PROCESSOR_VALUE_UNIT
    use: production
  profile: n3xc16.m48
  version: 10.0.5.4
  storageClassName: <default-storage-class>
```

b. Apply the YAML file:

```
oc apply -f <your_yaml_file>
```

c. To verify your API Connect cluster is successfully installed, run the following command:

```
oc get apic -n <APIC-namespace>
```

d. Verify that you can log in to the API Connect Cloud Manager UI:

To determine the location for logging in, view all the endpoints:

```
oc get routes -n <APIC-namespace>
```

e. Locate the `mgmt-admin-apic` endpoint, and access the Cloud Manager UI.

f. Login as the API Connect administrator.

When you install with the top-level CR, the password is auto-generated. To get the password:

```
oc get secret -n <APIC-namespace> | grep mgmt-admin-pass
oc get secret -n <APIC-namespace> <secret_name_from_previous_command> -o jsonpath="{.data.password}" | base64 -d &&
echo
```

10. Optional: Increase the timeout settings for the API Connect management endpoints, particularly for large deployments. See [Configuring timeouts for management endpoints](#).

What to do next

When you finish installing API Connect, [prepare your deployment for disaster recovery](#) so that your data can be restored in the event of an emergency.

Installing with a portable computer or storage device

You can use a portable device, such as a portal computer or a USB storage device, to perform an air-gapped installation of API Connect on Red Hat OpenShift Container Platform (OCP) when your cluster has no internet connectivity.

Before you begin

This task must be performed by a Red Hat OpenShift administrator.

About this task

If your cluster is not connected to the internet, you can mirror product images to a registry in your network-restricted environment by using a portable device. You can download images from the public registries on the internet to the portable device; then you can bring that portable device into the network-restricted environment (or transfer the files to a different device within the network-restricted environment) and mirror the images to the local registry. You can pull the images from the local registry to the target cluster for installation.

Note: In 10.0.5.3 and later, the API Connect operator is uncoupled from the IBM Cloud Pak foundational services operator. To accommodate the change, you will create environment variables and run commands to download and mirror two CASE files instead of one CASE file as in previous releases.

Procedure

1. Set up the mirroring environment.

a. Prepare the target cluster:

- Deploy a supported version of Red Hat OpenShift Container Platform (OCP) as a cluster.
For information, see Table 2 "API Connect and OpenShift Container Platform (OCP) compatibility matrix" in [IBM API Connect Version 10 software product compatibility requirements](#).
- Configure storage on the cluster and make sure that it is available.

b. Prepare the portable device:

You must be able to connect your portable device to the internet and to the restricted network environment (with access to the Red Hat OpenShift Container Platform (OCP) cluster and the local registry). The portable device must be on a Linux x86_64 or Mac platform with any operating system that the Red Hat OpenShift Client supports (in Windows, execute the actions in a Linux x86_64 VM or from a Windows Subsystem for Linux terminal).

- Ensure that the portable device has sufficient storage to hold all of the software that is to be transferred to the local registry.
- On the portable device, install either Docker or Podman (not both).
Docker and Podman are used for managing containers; you only need to install one of these applications.

- To install Docker (for example, on Red Hat Enterprise Linux), run the following commands:

```
yum check-update
yum install docker
```

- To install Podman, see the [Podman installation instructions](#).
For example, on Red Hat Enterprise Linux 9, install Podman with the following command:

```
yum install podman
```

iii. Install the Red Hat OpenShift Client tool (`oc`) as explained in [Getting started with the OpenShift CLI](#).

The `oc` tool is used for managing Red Hat OpenShift resources in the cluster.

iv. Download the IBM Catalog Management Plug-in for IBM Cloud Paks version 1.1.0 or later from [GitHub](#).

The `ibm-pak` plug-in enables you to access hosted product images, and to run `oc ibm-pak` commands against the cluster. To confirm that `ibm-pak` is installed, run the following command and verify that the response lists the command usage:

```
oc ibm-pak --help
```

c. Set up a local image registry and credentials.

The local Docker registry stores the mirrored images in your network-restricted environment.

i. Install a registry, or get access to an existing registry.

You might already have access to one or more centralized, corporate registry servers to store the API Connect images. If not, then you must install and configure a production-grade registry before proceeding.

The registry product that you use must meet the following requirements:

- Supports multi-architecture images through Docker Manifest V2, Schema 2
For details, see [Docker Manifest V2, Schema 2](#).
- Is accessible from the Red Hat OpenShift Container Platform cluster nodes
- Allows path separators in the image name

Note: Do not use the Red Hat OpenShift image registry as your local registry because it does not support multi-architecture images or path separators in the image name.

ii. Configure the registry to meet the following requirements:

- Supports auto-repository creation
- Has sufficient storage to hold all of the software that is to be transferred
- Has the credentials of a user who can create and write to repositories (the mirroring process uses these credentials)
- Has the credentials of a user who can read all repositories (the Red Hat OpenShift Container Platform cluster uses these credentials)

To access your registries during an air-gapped installation, use an account that can write to the target local registry. To access your registries during runtime, use an account that can read from the target local registry.

2. Set environment variables and download CASE files.

Create environment variables to use while mirroring images, connect to the internet, and download the API Connect CASE files.

a. Create the following environment variables with the installer image name and the image inventory on your portable device:

```
export CASE_NAME=ibm-apiconnect
export CASE_VERSION=4.0.5
export ARCH=amd64
```

For information on API Connect CASE versions and their corresponding operators and operands, see [Operator, operand, and CASE versions](#).

b. Connect your portable device to the internet (it does not need to be connected to the network-restricted environment at this time).

c. Download the CASE files to your portable device:

```
oc ibm-pak get $CASE_NAME --version $CASE_VERSION
```

If you omit the `--version` parameter, the command downloads the latest version.

3. Mirror the images.

The process of mirroring images pulls the images from the internet and pushes them to your local registry. After mirroring your images, you can configure your cluster and pull the images to it before installing API Connect.

a. Generate mirror manifests.

i. Define the environment variable `$TARGET_REGISTRY` by running the following command:

```
export TARGET_REGISTRY=<target-registry>
```

Replace `<target-registry>` with the IP address (or host name) and port of the local registry; for example: `172.16.0.10:5000`. If you want the images to use a specific namespace within the target registry, you can specify it here; for example: `172.16.0.10:5000/registry_ns`.

ii. Generate mirror manifests by running the following command:

```
oc ibm-pak generate mirror-manifests $CASE_NAME file://integration --version $CASE_VERSION --final-registry $TARGET_REGISTRY
```

If you need to filter for a specific image group, add the parameter `--filter <image_group>` to the command.

The `generate` command creates the following files at `~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION`:

- `catalog-sources.yaml`
- `catalog-sources-linux-<arch>.yaml` (if there are architecture-specific catalog sources)
- `image-content-source-policy.yaml`
- `images-mapping-to-filesystem.txt`
- `images-mapping-from-filesystem.txt`

The files are used when mirroring the images to the `TARGET_REGISTRY`.

b. Obtain an entitlement key for the entitled registry where the images are hosted:

- Log in to the [IBM Container Library](#).
- In the Container software library, select Get entitlement key.
- In the "Access your container software" section, click Copy key.
- Copy the key to a safe location; you will use it to log in to `cp.icr.io` in the next step.

c. Authenticate with the entitled registry where the images are hosted.

The image pull secret allows you to authenticate with the entitled registry and access product images.

i. Run the following command to export the path to the file that will store the authentication credentials that are generated on a Podman or Docker login:

```
export REGISTRY_AUTH_FILE=$HOME/.docker/config.json
```

The authentication file is typically located at `$HOME/.docker/config.json` on Linux or `%USERPROFILE%/.docker/config.json` on Windows.

ii. Log in to the `cp.icr.io` registry with Podman or Docker; for example:

```
podman login cp.icr.io
```

Use `cp` as the username and your entitlement key as the password.

d. Update the API Connect CASE manifest to correctly reference the DataPower Operator image.

Files for the DataPower Operator are now hosted on `icr.io`; however, the CASE manifest still refers to `docker.io` as the image host. To work around this issue, visit [Airgap install failure due to 'unable to retrieve source image docker.io'](#) in the DataPower documentation and update the manifest as instructed.

The manifest for API Connect (which uses the DataPower Operator) is stored in `~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION`.

After the manifest is updated, continue to the next step in this procedure.

e. Mirror the images from the internet to the portable device.

i. Define the environment variable `$IMAGE_PATH` by running the following command:

```
export IMAGE_PATH=<image-path>
```

where `<image-path>` indicates where the files will be stored on the portable device's file system.

ii. Mirror the images to the portable device:

```
oc image mirror \  
-f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping-to-filesystem.txt \  
--filter-by-os '*' \  
-a $REGISTRY_AUTH_FILE \  
--skip-multiple-scopes \  
--max-per-registry=1 \  
--dir "$IMAGE_PATH"
```

There might be a slight delay before you see a response to the command.

f. Move the portable device to the restricted-network environment.

The procedure depends on the type of device that you are using:

If you are using a portable computer, disconnect the device from the internet and connect it to the restricted-network environment. The same environment variables can be used.

If you are using portable storage, complete the following steps:

i. Transfer the following files to a device in the restricted-network environment:

- The `~/ibm-pak` directory.
- The contents of the `<image-path>` that you specified in the previous step.

ii. Create the same environment variables as on the original device; for example:

```
export CASE_NAME=ibm-apiconnect  
export CASE_VERSION=4.0.5  
export ARCH=amd64  
  
export REGISTRY_AUTH_FILE=$HOME/.docker/config.json  
  
export IMAGE_PATH=<image-path>
```

g. Authenticate with the local registry.

Log in to the local registry using an account that can write images to that registry; for example:

```
podman login $TARGET_REGISTRY
```

If the registry is insecure, add the following flag to the command: `--tls-verify=false`.

h. Mirror the product images to the target registry.

i. If you are using a portable computer, connect it to the restricted-network environment that contains the local registry.

If you are using portable storage, you already transferred files to a device within the restricted-network environment.

ii. Run the following command to copy the images to the local registry:

```
oc image mirror \  
-f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping-from-filesystem.txt \  
-a $REGISTRY_AUTH_FILE \  
--filter-by-os '*' \  
--skip-multiple-scopes \  
--max-per-registry=1 \  
--from-dir "$IMAGE_PATH"
```

Note: If the local registry is not secured by TLS, or the certificate presented by the local registry is not trusted by your device, add the `--insecure` option to the command.

There might be a slight delay before you see a response to the command.

i. Configure the target cluster.

Now that images have been mirrored to the local registry, the target cluster must be configured to pull the images from it. Complete the following steps to configure the cluster's global pull secret with the local registry's credentials and then instruct the cluster to pull the images from the local registry.

i. Log in to your Red Hat OpenShift Container Platform cluster:

```
oc login <openshift_url> -u <username> -p <password> -n <namespace>
```

ii. [Update the global image pull secret](#) for the cluster as explained in the Red Hat OpenShift Container Platform documentation.

Note: If you have an insecure registry, add the registry to the cluster's `insecureRegistries` list by running the following command:

```
oc edit image.config.openshift.io/cluster -o yaml
```

and add the `TARGET_REGISTRY` to `spec.registrySources.insecureRegistries` as shown in the following example:

```
spec:  
  registrySources:  
    insecureRegistries:  
      - insecure0.svc:5001  
      - <TARGET_REGISTRY>
```

If the `insecureRegistries` field does not exist, you can add it.

iii. Create the `ImageContentSourcePolicy`, which instructs the cluster to pull the images from your local registry (run both commands):

```
oc apply -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/image-content-source-policy.yaml
```

iv. Verify that each `ImageContentSourcePolicy` resource was created:

```
oc get imageContentSourcePolicy
```

v. Verify your cluster node status:

```
oc get MachineConfigPool -w
```

Wait for all nodes to be updated before proceeding to the next step.

4. Apply the catalog sources.

Now that you have mirrored images to the target cluster, apply the catalog sources.

In the following steps, replace **<Architecture>** with either **amd64**, **s390x** or **ppc64le** as appropriate for your environment.

a. Export the variables for the command line to use:

```
export CASE_NAME=ibm-apiconnect
export CASE_VERSION=4.0.5
export ARCH=amd64
```

b. Generate the catalog sources and save them in another directory in case you need to replicate this installation in the future.

i. Get the catalog sources:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
```

ii. Get any architecture-specific catalog sources that you need to back up as well:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

You can also navigate to the directory in your file browser to copy these artifacts into files that you can keep for re-use or for pipelines.

c. Apply the catalog sources to the cluster.

i. Apply the universal catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
```

ii. Apply any architecture-specific catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

iii. Confirm that the catalog sources were created in the **openshift-marketplace** namespace:

```
oc get catalogsource -n openshift-marketplace
```

5. Create the namespace where you will install API Connect.

a. Specify the namespace where you want to install the operator:

```
export NAMESPACE=<APIC-namespace>
```

The namespace where you install API Connect must meet the following requirements:

- Red Hat OpenShift Container Platform (OCP): Only one top-level CR (APIConnectCluster) can be deployed in each namespace.
- Cloud Pak for Integration: Only one API Connect capability can be deployed in each namespace.
- The following namespaces cannot be used to install API Connect because Red Hat OpenShift Container Platform (OCP) restricts the use of default namespaces for installing non-cluster services:
 - **default**
 - **kube-system**
 - **kube-public**
 - **openshift-node**
 - **openshift-infra**
 - **openshift**

b. Create a new namespace for installing the operator:

```
oc new-project $NAMESPACE
```

c. Create an **OperatorGroup**:

i. Create a YAML file called **apiconnect-operator-group.yaml** similar to the following example, replacing **<APIC-namespace>** with your new namespace:

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: ibm-apiconnect-operatorgroup
spec:
  targetNamespaces:
  - <APIC-namespace>
```

ii. Add the new operator group to your namespace:

```
oc apply -f apiconnect-operator-group.yaml -n ${NAMESPACE}
```

6. (10.0.5.3 and later) Create the **ibm-common-services-operator** subscription, if it does not already exist.

This step only applies when you install version 10.0.5.3 or later. If you are installing an older version of API Connect, skip this step.

The **ibm-common-services-operator** is provided by IBM Cloud Pak foundational services.

a. Create a file called **common-services-sub.yaml** and paste in the following contents:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-common-service-operator
  namespace: <APIC-namespace>
spec:
  channel: <channel>
  installPlanApproval: Automatic
```

```
name: ibm-common-service-operator
source: opencloud-operators
sourceNamespace: openshift-marketplace
```

Where:

- `<APIC-namespace>` the namespace where you will install API Connect.
- `<channel>` is one of the following values:
 - `v3.23` if you are using IBM Cloud Pak foundational services for Continuous Delivery
 - `v3` if you are using IBM Cloud Pak foundational services for Long Term Service Release

b. Apply the subscription with the following command:

```
oc apply -f common-services-sub.yaml
```

c. Select Operators, Installed Operators, and ensure that Project: All Projects is selected.

If any operators such as `ibm-apiconnect` or `ibm-cert-manager-operator` show the status of "Upgrade available", approve the upgrade by completing the following steps:

- Click Upgrade available.
- Click Preview InstallPlan.
- Click Approve.
- Check the `ibm-common-services` namespace and ensure that all operators with a status of "Upgrade available" are approved.
- Wait for the `IBM Cloud Pak foundational services, IBM NamespaceScope`, and `Operand Deployment Lifecycle Manager` operators to install.

7. (10.0.5.3 and later) Install cert manager:

This step only applies when you install version 10.0.5.3 or later. If you are installing an older version of API Connect, skip this step.

a. Create a file called `cert-manager-operand-request.yaml` and paste in the following content:

```
apiVersion: operator.ibm.com/v1alpha1
kind: OperandRequest
metadata:
  name: ibm-apiconnect-cert-manager
  namespace: <APIC-namespace>
spec:
  requests:
  - operands:
    - name: ibm-cert-manager-operator
      registry: common-service
      registryNamespace: ibm-common-services
```

b. Create the `operandRequest` for cert-manager by running the following command:

```
oc apply -f cert-manager-operand-request.yaml
```

c. Select Operators, Installed Operators, and ensure that Project: All Projects is selected.

Wait for the `ibm-cert-manager-operator` to display and if it shows the status of "Upgrade available", approve the upgrade by completing the following steps:

- Click Upgrade available.
- Click Preview InstallPlan.
- Click Approve.
- Wait for the `IBM Cert Manager` operator to install.

8. Create a Subscription for the IBM

`APICConnect` operator.

a. Create a YAML file called `apic-sub.yaml` similar to the following example:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-apiconnect
spec:
  channel: v3.4
  name: ibm-apiconnect
  source: ibm-apiconnect-catalog
  sourceNamespace: openshift-marketplace
```

b. Apply the new subscription to your namespace:

```
oc apply -f apic-sub.yaml -n ${NAMESPACE}
```

9. Install API Connect (the operand).

API Connect provides one top-level CR that includes all of the API Connect subsystems (Management, Developer Portal, Analytics, and Gateway).

Tip: Before you install API Connect, you can customize the configuration of the analytics subsystem. Review the information in [Planning your analytics deployment](#) to choose configuration options, and then configure the settings as explained in [Analytics subsystem CR configured settings](#). If you are installing with the top-level `APICConnectClusterCR`, add your configuration settings to the `spec.analytics` section of the CR; if you are installing the subsystems individually, make your changes to the `AnalyticsCluster` CR.

a. Create a YAML file to use for deploying the top-level `APICConnectCluster` CR. Use the template that applies to your deployment (non-production or production).

Note: The values shown in the following examples might not be suitable for your deployment. For information on the license, profile, and version settings, as well as additional configuration settings, see [API Connect configuration settings](#).

- Example CR settings for a one replica deployment:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APICConnectCluster
metadata:
  labels:
    app.kubernetes.io/instance: apiconnect
```

```

app.kubernetes.io/managed-by: ibm-apiconnect
app.kubernetes.io/name: apiconnect-minimum
name: <name_of_your_instance>
namespace: <APIC-namespace>
spec:
  license:
    accept: true
    license: L-GVEN-GFUPVE
    metric: PROCESSOR_VALUE_UNIT
    use: nonproduction
  profile: nlxc17.m48
  version: 10.0.5.4
  storageClassName: <default-storage-class>

```

- Example CR settings for a three replica deployment:

```

apiVersion: apiconnect.ibm.com/v1beta1
kind: APICConnectCluster
metadata:
  labels:
    app.kubernetes.io/instance: apiconnect
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: apiconnect-production
  name: <name_of_your_instance>
  namespace: <APIC-namespace>
spec:
  license:
    accept: true
    license: L-GVEN-GFUPVE
    metric: PROCESSOR_VALUE_UNIT
    use: production
  profile: n3xc16.m48
  version: 10.0.5.4
  storageClassName: <default-storage-class>

```

b. Apply the YAML file:

```
oc apply -f <your_yaml_file>
```

c. To verify your API Connect cluster is successfully installed, run the following command:

```
oc get apic -n <APIC-namespace>
```

d. Verify that you can log in to the API Connect Cloud Manager UI:
To determine the location for logging in, view all the endpoints:

```
oc get routes -n <APIC-namespace>
```

e. Locate the `mgmt-admin-apic` endpoint, and access the Cloud Manager UI.

f. Login as the API Connect administrator.

When you install with the top-level CR, the password is auto-generated. To get the password:

```

oc get secret -n <APIC-namespace> | grep mgmt-admin-pass
oc get secret -n <APIC-namespace> <secret_name_from_previous_command> -o jsonpath="{.data.password}" | base64 -d &&
echo

```

10. Optional: Increase the timeout settings for the API Connect management endpoints, particularly for large deployments. See [Configuring timeouts for management endpoints](#).

What to do next

When you finish installing API Connect, [prepare your deployment for disaster recovery](#) so that your data can be restored in the event of an emergency.

Enable additional features post-install

After installing API Connect, you can enable additional features that are related to inter-subsystem communication.

- [Enable management CA verification on REST API calls](#)
Enable the portal and gateway to validate the management subsystem's REST API server certificates.
- [Enable JWT security and disable mTLS between subsystems](#)
Postinstallation steps to enable JWT security and disable mTLS between subsystems.

Enable management CA verification on REST API calls

Enable the portal and gateway to validate the management subsystem's REST API server certificates.

About this task

After you install API Connect, you can enable the portal and gateway to verify the REST API server certificate when they make calls to the REST API. The gateway and portal subsystems make calls to the management subsystem REST API during their normal operations. The gateway makes calls to the platform REST API, and the portal makes calls to the platform REST API and the consumer REST API. To verify the CA certificates of the REST API endpoints, enter the secret names of the CA certificates into the portal and gateway subsystem sections of the top-level CR.

Note:

If you are not sure of the secret names, follow these steps to determine their names:

1. Check the management subsystem CR with `oc describe mgmt`, and identify the issuer of the platform and consumer API endpoints:

```
oc describe mgmt -n <namespace>

...
Platform API Endpoint:
  Annotations:
    cert-manager.io/issuer:          <instance name>-ingress-issuer
...
Consumer API Endpoint:
  Annotations:
    cert-manager.io/issuer:          <instance name>-ingress-issuer
```

2. Describe the issuer with `oc describe issuer`, and identify the secret name:

```
oc describe issuer <instance name>-ingress-issuer -n <namespace>
...
Spec:
  Ca:
    Secret Name: <instance name>-ingress-ca
...

```

Note: If you are using `in-cluster` communication (see [In-cluster service communication between subsystems](#)), then the portal and gateway make REST calls on the service endpoints, instead of the `external` endpoints. In the steps documented in this topic, replace `mgmtPlatformEndpointCASecret` with `mgmtPlatformEndpointSvcCASecret`, and `mgmtConsumerEndpointCASecret` with `mgmtConsumerEndpointSvcCASecret`. For `secretName`, the default for the service endpoints is `mgmt-ca` or `<apic instance name>-mgmt-ca`.

Procedure

1. Edit the top-level CR, `apiconnectcluster`:

```
oc edit apiconnectcluster -n <namesapce>
```

2. In the `spec`: section, insert a portal section with the CA secret names for the platform and consumer REST APIs:

```
spec:
...
  portal:
    mgmtPlatformEndpointCASecret:
      secretName: <instance name>-ingress-ca
    mgmtConsumerEndpointCASecret:
      secretName: <instance name>-ingress-ca
```

Note: If the portal section already exists, insert the `mgmtPlatformEndpointCASecret` and `mgmtConsumerEndpointCASecret` sections inside it.

3. In the `spec`: section, insert a gateway section with the CA secret name for the platform REST API:

```
spec:
...
  gateway:
    mgmtPlatformEndpointCASecret:
      secretName: <instance name>-ingress-ca
```

Note: If the gateway section already exists, insert the `mgmtPlatformEndpointCASecret` section inside it.

4. Check the contents of the `ptl` and `gw` CRs to confirm that the change is applied.

```
oc get ptl -o yaml -n <namesapce>

...
  mgmtPlatformEndpointCASecret:
    secretName: <instance name>-ingress-ca
  mgmtConsumerEndpointCASecret:
    secretName: <instance name>-ingress-ca
...

oc get gw -o yaml -n <namesapce>

...
  mgmtPlatformEndpointCASecret:
    secretName: <instance name>-ingress-ca
...

```

Enable JWT security and disable mTLS between subsystems

Postinstallation steps to enable JWT security and disable mTLS between subsystems.

About this task

JWT security is an alternative to using mTLS to secure inter-subsystem communication. For more information on JWT security and when to use it, see: [Enable JWT security instead of mTLS](#).

If you disable mTLS then you must enable JWT.

Note: It is not possible to use JWT on the V5 compatible gateway to analytics message flow. This flow is secured by mTLS, which cannot be disabled.

Procedure

1. Describe the `apiconnectcluster` CR to get the JWKS URL:

```
oc describe apiconnectcluster -n <namespace>

status:
- name: jwksUrl
  secretName: api-endpoint
  type: API
  uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

2. For each portal, gateway, and analytics subsystem where you want to use JWT security, edit the section of the top-level CR corresponding to that subsystem, and add the following:

```
spec:
  <subsystem>
    mtlsValidateClient: false
    jwksUrl: <JWKS URL>
```

where

- `<subsystem>` is either `portal`, `analytics`, or `gateway`, depending on which subsystem you want to use JWT security with. If this subsystem section does not already exist in the top-level CR, then add it as shown above.
 - `<JWKS URL>` is the URL identified in step 1.
3. **v10.0.5.4 and later:** Enable JWT on the gateway to analytics communications flow. Enable the Use JWT switch for the registered gateway in the Topology page of the Cloud Manager UI.

DataPower version 10.5.0.4 only: To enable JWT on the gateway to analytics communications flow, you also must add a `dataPowerOverride` section to the gateway CR:

```
spec:
  jwksUrl: <JWKS URL>
  dataPowerOverride:
    image: customregistry.com/custom-image-datapower:10.5.0.5
    version: 10.5.0.5
    license: X-XXXX-XXXXXX
```

Installing with subsystem CRs in a shared namespace on OpenShift

Use the individual subsystem CRs to install API Connect subsystems in a shared namespace on OpenShift.

About this task

API Connect subsystems can be installed by creating individual subsystem CRs in the same shared namespace, instead of using a top-level CR. If you want to install your API Connect subsystems in different namespaces, for example Management server in a namespace that is called `mgmt`, and gateway in a namespace that is called `gway`, then see: [Installing with subsystem CRs in different namespaces or environments on OpenShift](#).

1. [Installing operators](#)
Install operators for API Connect, DataPower, and common services so that you can deploy the API Connect subsystems in a shared namespace on OpenShift.
2. [Setting up a certificate issuer](#)
Create the cert-manager issuer for generating the certificates and secrets for use with API Connect.
3. [Installing the Management subsystem in a shared namespace](#)
Install the Management subsystem by creating and applying the `mgmt_cr.yaml` file.
4. [Installing the Gateway subsystem in a shared namespace](#)
Install the Gateway subsystem by creating and applying the `apigateway_cr.yaml` file.
5. [Installing the Portal subsystem in a shared namespace](#)
Install the Portal subsystem by creating and applying the `portal_cr.yaml` file.
6. [Installing the Analytics subsystem in a shared namespace](#)
Install the analytics subsystem by creating and applying the `analytics_cr.yaml` file.

Installing operators

Install operators for API Connect, DataPower, and common services so that you can deploy the API Connect subsystems in a shared namespace on OpenShift.

Before you begin

The API Connect operator and operand must be from the same release and fix pack level. Table 1 lists the current version of the operator and operand for API Connect.

Table 1. API Connect operator and operand versions

API Connect (operand)	Operator channel version	Highest operator version
10.0.5.4	v3.4	3.4.0

You can use any combination of API Connect 10.0.5.x with DataPower Gateway 10.5.0.x or DataPower API Gateway 10.5.0.x.

Procedure

1. Obtain an entitlement key for the Entitled Registry:

- a. Log in to the [IBM Container Library](#).
 - b. In the Container software library, select Get entitlement key.
 - c. After the Access your container software heading, click Copy key.
 - d. Copy the key to a safe location.
2. Create the namespace where API Connect will be installed, either by selecting Home->Projects->Create Project in the UI, or with the command:

```
oc create ns <APIC-namespace>
```

The namespace where you install API Connect must meet the following requirements:

- OpenShift restricts the use of default namespaces for installing non-cluster services. The following namespaces cannot be used to install API Connect: **default, kube-system, kube-public, openshift-node, openshift-infra, openshift.**
 - OpenShift: Only one top-level CR (APIConnectCluster) can be deployed in each namespace.
 - Cloud Pak for Integration: Only one API Connect capability can be deployed in each namespace.
3. If you are installing the operator into a single namespace, create an OperatorGroup object specifying that namespace.

Attention: If you are installing the operator into all namespaces, skip this step because the **openshift-operators** namespace already has an appropriate Operator group in place.

- a. Create the apiconnect-operator-group.yaml file with an OperatorGroup object that identifies the namespace in which RBAC permissions for the IBM API Connect Operator will be generated, and where the CSV will be available:

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: <operator-group-name>
  namespace: <APIC-namespace>
spec:
  targetNamespaces:
  - <APIC-namespace>
```

- b. Apply the OperatorGroup object with the following command:

```
oc apply -f apiconnect-operator-group.yaml
```

4. Create the image pull secret in the namespace where you want to install API Connect:

- a. Open the OpenShift web console, click Workloads, > Secrets.
- b. Ensure that the Project is set to the namespace where you intend to install API Connect.
- c. Click Create and select Image pull secret.
- d. Set the following parameters for the secret:
 - Set Secret name to **ibm-entitlement-key**.
 - Set Authentication type to **Image registry credentials**.
 - Set Registry server address to **cp.icr.io**.
 - Set Username to **cp**.
 - Set Password to the entitlement key generated in Step 1.
 - Click Create to create the secret.

Note: If your OpenShift platform is Red Hat OpenShift on IBM Cloud (ROKS), you might need to reload your worker nodes after the image pull secret is created. You can reload the worker nodes either from the OpenShift web console or from the command line with: **oc worker reload**.

5. Add the IBM Operator Catalog source:

- a. In the OpenShift web console, and click + (in the page banner) to open the Import YAML dialog box.
- b. Paste the following resource definition into the dialog box:

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: ibm-operator-catalog
  namespace: openshift-marketplace
spec:
  displayName: "IBM Operator Catalog"
  publisher: IBM
  sourceType: grpc
  image: icr.io/cpopen/ibm-operator-catalog
  updateStrategy:
    registryPoll:
      interval: 45m
```

- c. Click Create.

6. Create the apiconnect subscription.

- a. Create the apiconnect subscription with the appropriate channel by creating a file called **apic-sub.yaml** and pasting in the following contents, updating the namespace as required:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-apiconnect
  namespace: <namespace>
spec:
  channel: v3.4
  name: ibm-apiconnect
  source: ibm-operator-catalog
  sourceNamespace: openshift-marketplace
```

Where:

- **<namespace>** is set to **openshift-operators** if you are installing the operator in all namespaces. Otherwise set this to the namespace created in [step 2](#).

- b. Apply the subscription with the following command:

```
oc apply -f apic-sub.yaml
```


- c. Select Operators, Installed Operators, and ensure that Project: All Projects is selected.
If any operators such as `ibm-apiconnect` or `ibm-cert-manager-operator` show the status of "Upgrade available", approve the upgrade by completing the following steps:
- Click Upgrade available.
 - Click Preview InstallPlan.
 - Click Approve.
 - Wait for the **IBM API Connect** and **IBM DataPower Gateway** operators to install.

The **IBM DataPower Gateway** operator is a prerequisite to API Connect and must not be removed.

7. (10.0.5.3 and later) Create the `ibm-common-services-operator` subscription, if it does not already exist.
This step only applies when you install version 10.0.5.3 or later. If you are installing an older version of API Connect, skip this step.

- a. Create a file called `common-services-sub.yaml` and paste in the following contents:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-common-service-operator
  namespace: <namespace>
spec:
  channel: <channel>
  name: ibm-common-service-operator
  source: ibm-operator-catalog
  sourceNamespace: openshift-marketplace
```

Where:

- `<namespace>` is one of the following values:
 - `openshift-operators` if you are installing the operator in all namespaces
 - The namespace created in step 2 if you are installing the operator in a single namespace
- `<channel>` is one of the following values:
 - `v3.23` if you are using IBM Cloud Pak foundational services for CD (continuous delivery)
 - `v3` if you are using IBM Cloud Pak foundational services for LTSR (long term service release)

- b. Apply the subscription with the following command:

```
oc apply -f common-services-sub.yaml
```

- c. Select Operators, Installed Operators, and ensure that Project: All Projects is selected.
If any operators such as `ibm-apiconnect` or `ibm-cert-manager-operator` show the status of "Upgrade available", approve the upgrade by completing the following steps:
- Click Upgrade available.
 - Click Preview InstallPlan.
 - Click Approve.
 - Check the `ibm-common-services` namespace and ensure that all operators with a status of "Upgrade available" are approved.
 - Wait for the **IBM Cloud Pak foundational services**, **IBM NamespaceScope**, and **Operand Deployment Lifecycle Manager** operators to install.

8. (10.0.5.3 and later) Install cert manager:

This step only applies when you install version 10.0.5.3 or later. If you are installing an older version of API Connect, skip this step.

- a. Create a file called `cert-manager-operand-request.yaml` and paste in the following content:

```
apiVersion: operator.ibm.com/v1alpha1
kind: OperandRequest
metadata:
  name: ibm-apiconnect-cert-manager
  namespace: <namespace>
spec:
  requests:
  - operands:
    - name: ibm-cert-manager-operator
      registry: common-service
      registryNamespace: ibm-common-services
```

- b. Create the `operandRequest` for cert-manager by running the following command:

```
oc apply -f cert-manager-operand-request.yaml
```

- c. Select Operators, Installed Operators, and ensure that Project: All Projects is selected.
Wait for the `ibm-cert-manager-operator` to display and if it shows the status of "Upgrade available", approve the upgrade by completing the following steps:
- Click Upgrade available.
 - Click Preview InstallPlan.
 - Click Approve.
 - Wait for the **IBM Cert Manager** operator to install.

Next topic: [Setting up a certificate issuer](#)

Setting up a certificate issuer

Create the cert-manager issuer for generating the certificates and secrets for use with API Connect.

Before you begin

To deploy API Connect, you must first create the cert-manager issuers and certificates that API Connect uses. On OpenShift cert-manager is bundled as a foundational service, so you do not need to install cert-manager itself. For more information on cert-manager: [Key Concepts: Cert-manager, Issuers, and Secrets](#)

Procedure

1. Create a file that is called `ingress-issuer-v1.yaml` and paste in the following contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
---

apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "selfsigning-issuer"
  }
spec:
  selfSigned: {}
---

apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ingress-ca
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-ca"
  }
spec:
  duration: 87600h # 10 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretName: ingress-ca
  commonName: "ingress-ca"
  usages:
  - digital signature
  - key encipherment
  - cert sign
  isCA: true
  issuerRef:
    name: selfsigning-issuer
    kind: Issuer
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "management"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "ingress-ca"
---

apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-issuer"
  }
spec:
  ca:
    secretName: ingress-ca
---

apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: portal-admin-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "portal-admin-client"
  }
spec:
  subject:
    organizations:
    - cert-manager
```

```

commonName: portal-admin-client
secretName: portal-admin-client
issuerRef:
  name: ingress-issuer
usages:
- "client auth"
- "signing"
- "key encipherment"
duration: 17520h # 2 years
renewBefore: 720h # 30 days
privateKey:
  rotationPolicy: Always
secretTemplate:
  labels:
    app.kubernetes.io/instance: "management"
    app.kubernetes.io/managed-by: "ibm-apiconnect"
    app.kubernetes.io/name: "portal-admin-client"
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-client-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-client-client"
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: gateway-client-client
  secretName: gateway-client-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "management"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "gateway-client-client"
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: analytics-ingestion-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "analytics-ingestion-client"
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: analytics-ingestion-client
  secretName: analytics-ingestion-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "management"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "analytics-ingestion-client"
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-service
  labels: {
    app.kubernetes.io/instance: "gatewaycluster",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-service"
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: gateway-service
  secretName: gateway-service
  issuerRef:

```

```

    name: ingress-issuer
    usages:
    - "client auth"
    - "signing"
    - "key encipherment"
    duration: 17520h # 2 years
    renewBefore: 720h # 30 days
    privateKey:
      rotationPolicy: Always
    secretTemplate:
      labels:
        app.kubernetes.io/instance: "gatewaycluster"
        app.kubernetes.io/managed-by: "ibm-apiconnect"
        app.kubernetes.io/name: "gateway-service"
  ---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-peering
  labels: {
    app.kubernetes.io/instance: "gatewaycluster",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-peering"
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: gateway-peering
  secretName: gateway-peering
  issuerRef:
    name: ingress-issuer
  usages:
  - "server auth"
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "gatewaycluster"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "gateway-peering"

```

2. Apply the file to your namespace with `oc apply -f ingress-issuer-v1.yaml -n <apic_namespace>`
3. Verify that the command installation succeeded by running the following command:

```
oc get certificates -n <apic_namespace>
```

All certificates created successfully:

NAME	READY	SECRET	AGE
analytics-ingestion-client	True	analytics-ingestion-client	70s
gateway-peering	True	gateway-peering	69s
gateway-service	True	gateway-service	69s
ingress-ca	True	ingress-ca	71s
portal-admin-client	True	portal-admin-client	71s

Previous topic: [Installing operators](#)

Next topic: [Installing the Management subsystem in a shared namespace](#)

Installing the Management subsystem in a shared namespace

Install the Management subsystem by creating and applying the `mgmt_cr.yaml` file.

Before you begin

Complete the following tasks to prepare for deploying API Connect:

1. [Preparing for installation](#)
2. [Installing operators](#)
3. [Setting up a certificate issuer](#)

About this task

Use the OpenShift CLI to edit the custom resource template for the Management subsystem then apply the resource. Verify that the pods are up and running, and validate that you can connect to the API Connect Cloud Manager.

Procedure

1. Create a file that is called `mgmt_cr.yaml` and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
```

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementCluster
metadata:
  name: management
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "management"
  }
spec:
  version: $APP_PRODUCT_VERSION
  profile: $PROFILE
  portal:
    admin:
      secretName: portal-admin-client
  analytics:
    ingestion:
      secretName: analytics-ingestion-client
  gateway:
    client:
      secretName: gateway-client-client
  cloudManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: admin.$STACK_HOST
        secretName: cm-endpoint
  apiManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: manager.$STACK_HOST
        secretName: apim-endpoint
  platformAPIEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: api.$STACK_HOST
        secretName: api-endpoint
  consumerAPIEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: consumer.$STACK_HOST
        secretName: consumer-endpoint
  databaseVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
  microServiceSecurity: certManager
  certManagerIssuer:
    name: selfsigning-issuer
    kind: Issuer
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
```

2. Edit the yaml file and replace the variables:

\$APP_PRODUCT_VERSION
API Connect application version for the subsystems.

version: <version_number>

Example version number: 10.0.5.4

\$PROFILE

Specify your Management subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see [Planning your deployment topology](#).

\$STACK_HOST

The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can do one of the following:

- Subdomain customization only
Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **STACK_HOST** to be the desired ingress subdomain for the API Connect stack. For example, if your host is **myhost.subnet.example.com**:

```
cloudManagerEndpoint:
  < ... >
  hosts:
```

```

- name: admin.myhost.subnet.example.com
  secret: cm-endpoint

apiManagerEndpoint:
  < ... >
  hosts:
  - name: manager.myhost.subnet.example.com
    secret: apim-endpoint

platformAPIEndpoint:
  < ... >
  hosts:
  - name: api.myhost.subnet.example.com
    secret: api-endpoint

consumerAPIEndpoint:
  < ... >
  hosts:
  - name: consumer.myhost.subnet.example.com
    secret: consumer-endpoint

```

- Complete hostname customization
Change both the predefined prefixes and the `STACK_HOST` subdomain to match your desired hostnames.

For example, for `cloudManagerEndpoint`, you can replace `admin.$STACK_HOST` with `my.cloudmgr.myhost.subnet.example.com`, where `my.cloudmgr` replaces `admin`, and `myhost.subnet.example.com` replaces `STACK_HOST`. For example:

```

cloudManagerEndpoint:
  < ... >
  hosts:
  - name: my.cloudmgr.myhost.subnet.example.com
    secret: cm-endpoint

```

You can do this for some or all of the hostnames, depending on your customization requirements.

`$STORAGE_CLASS`

The OCP storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: `oc get sc`.

```
storageClassName: local-storage
```

`$LICENSE_ACCEPTANCE`

Set `accept` to `true`. You must accept the license to successfully deploy API Connect.

`$LICENSE_USE`

Set `use` to either `production` or `nonproduction` to match the license that you purchased.

`$LICENSE_ID`

Set `license` to the license ID for the version of API Connect that you purchased. See [API Connect licenses](#).

3. Configure database backups for the Management subsystem.

To successfully restore API Connect in the event of a disaster where you must redeploy a new cluster, you must configure the database backup settings before the initial installation of the Management subsystem. For instructions on configuring backup settings, see [Configuring backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration](#).

4. Install the Management subsystem by applying the modified CR with the following command:

```
oc apply -f mgmt_cr.yaml -n <management_namespace>
```

5. Verify that the Management subsystem is fully installed by running the following command:

```
oc get ManagementCluster -n <management_namespace>
```

The installation is complete when the `READY` status is `True`, and the `SUMMARY` reports that all services are online:

NAME	READY	SUMMARY	VERSION	RECONCILED	VERSION	AGE
management	True	16/16	<version>		<version-build>	7m17s

6. Verify that you can log in to the API Connect Cloud Manager UI:

- a. Determine the URL for the Cloud Manager UI by running the following command to view the API Connect endpoints:

```
oc get routes -n <management_namespace>
```

- b. Locate the `management-admin` endpoint, and note it down.

- c. Determine the Cloud Manager administrator password by running the following commands:

```
oc get secret -n <management-namespace> | grep management-admin-secret
```

```
oc get secret -n <management-namespace> <secret_name_from_previous_command> -o jsonpath="{.data.password}" | base64 -d && echo ""
```

- d. Open a browser and go to the `management-admin-apic` endpoint.

- e. Log in to the Cloud Manager as `admin` with the administrator password.

7. Store all secrets and passwords in a safe place.

8. Prepare the Management subsystem for disaster recovery.

Follow the appropriate instructions for your environment:

- [Preparing the Management subsystem for disaster recovery on OpenShift](#)
- [Preparing the Management subsystem for disaster recovery on Cloud Pak for Integration](#)

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the [Cloud Manager configuration checklist](#).

Previous topic: [Setting up a certificate issuer](#)

Next topic: [Installing the Gateway subsystem in a shared namespace](#)

Installing the Gateway subsystem in a shared namespace

Install the Gateway subsystem by creating and applying the `apigateway_cr.yaml` file.

Before you begin

Complete the following tasks to prepare for deploying API Connect:

1. [Preparing for installation](#)
2. [Installing operators](#)
3. [Setting up a certificate issuer](#)

About this task

Use the OpenShift CLI to edit the custom resource template for the Gateway, apply the resource, verify that the pods are up and running.

Procedure

1. Create the gateway admin secret with: `oc -n <gateway namespace> create secret generic admin-secret --from-literal=password=admin`
2. If you are installing the API Gateway (v6), create a yaml file that is called `apigateway_cr.yaml` and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

apiVersion: gateway.apiconnect.ibm.com/v1beta1
kind: GatewayCluster
metadata:
  name: gwv6
  labels: {
    app.kubernetes.io/instance: "gateway",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gwv6"
  }
spec:
  version: $APP_PRODUCT_VERSION
  profile: $PROFILE
  apicGatewayServiceV5CompatibilityMode: false
  mgmtPlatformEndpointCASecret:
    secretName: ingress-ca
  mgmtPlatformEndpointSvcCASecret:
    secretName: management-ca
  gatewayEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: rgw.$STACK_HOST
        secretName: gwv6-endpoint
  gatewayManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: rgwd.$STACK_HOST
        secretName: gwv6-manager-endpoint
  apicGatewayServiceTLS:
    secretName: gateway-service
  apicGatewayPeeringTLS:
    secretName: gateway-peering
  datapowerLogLevel: 3
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
  tokenManagementService:
    enabled: true
```

```

storage:
  storageClassName: $STORAGE_CLASS
  volumeSize: 30Gi
adminUser:
  secretName: admin-secret
# syslogConfig:
# enabled: false # if true, provide below details
# remoteHost: $DATAPOWER_SYSLOG_TCP_REMOTE_HOST # must be a string
# remotePort: $DATAPOWER_SYSLOG_TCP_REMOTE_PORT # must be an int
# secretName: $DATAPOWER_SYSLOG_TCP_TLS_SECRET # must be a string

```

If you are installing the v5 Gateway, create a yaml file that is called v5gateway_cr.yaml and paste in these contents:

```

#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

apiVersion: gateway.apiconnect.ibm.com/v1beta1
kind: GatewayCluster
metadata:
  name: gvw5
  labels: {
    app.kubernetes.io/instance: "gateway",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gvw5"
  }
spec:
  version: $APP_PRODUCT_VERSION
  profile: $PROFILE
  apicGatewayServiceV5CompatibilityMode: true
  gatewayEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: gw.$STACK_HOST
        secretName: gvw5-endpoint
  gatewayManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: gwd.$STACK_HOST
        secretName: gvw5-manager-endpoint
  apicGatewayServiceTLS:
    secretName: gateway-service
  apicGatewayPeeringTLS:
    secretName: gateway-peering
  datapowerLogLevel: 3
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
  adminUser:
    secretName: admin-secret
# syslogConfig:
# enabled: false # if true, provide below details
# remoteHost: $DATAPOWER_SYSLOG_TCP_REMOTE_HOST # must be a string
# remotePort: $DATAPOWER_SYSLOG_TCP_REMOTE_PORT # must be an int
# secretName: $DATAPOWER_SYSLOG_TCP_TLS_SECRET # must be a string

```

3. Edit the yaml file and set the variables:

\$APP_PRODUCT_VERSION
API Connect application version for the subsystems.

```
version: <version_number>
```

Example version number: 10.0.5.4

\$PROFILE

Specify your Gateway subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see [Planning your deployment topology](#).

\$STACK_HOST

The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can do one of the following:

- Subdomain customization only
Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **STACK_HOST** to be the desired ingress subdomain for the API Connect stack. For example, if your host is **myhost.subnet.example.com**:

```

gatewayEndpoint:
  < ... >
  hosts:
    - name: rgw.myhost.subnet.example.com

```



```

    secret: gwv6-endpoint

gatewayManagerEndpoint:
  < ... >
  hosts:
  - name: rgwd.myhost.subnet.example.com
    secret: gwv6-manager-endpoint

```

- Complete hostname customization
Change both the predefined prefixes and the `STACK_HOST` subdomain to match your desired hostnames.

For example, for `gatewayEndpoint`, you can replace `rgw.$STACK_HOST` with `my.gateway.endpoint.myhost.subnet.example.com`, where `my.gateway.endpoint` replaces `rgw`, and `myhost.subnet.example.com` replaces `STACK_HOST`:

```

gatewayEndpoint:
  < ... >
  hosts:
  - name: my.gateway.endpoint.myhost.subnet.example.com
    secret: gwv6-endpoint

```

You can do this for some or all of the hostnames, depending on your customization requirements.

`$STORAGE_CLASS`

The OCP storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: `oc`

```

get
sc.

```

```

storageClassName: local-storage

```

`$LICENSE_ACCEPTANCE`

Set `accept` to `true`. You must accept the license to successfully deploy API Connect.

`$LICENSE_USE`

Set `use` to either `production` or `nonproduction` to match the license that you purchased.

`$LICENSE_ID`

Set `license`: to the license ID for the version of API Connect that you purchased. See [API Connect licenses](#).

`syslogConfig`

To enable this feature uncomment this section, set `enabled`: `true`, and set the properties:

`$DATAPOWER_SYSLOG_TCP_REMOTE_HOST`

```

remoteHost: <remote.host>

```

String. The hostname of the server for `syslogConfig`.

`$DATAPOWER_SYSLOG_TCP_REMOTE_PORT`

```

remotePort: <remote.port.number>

```

Integer. The port number of the server for `syslogConfig`.

`$DATAPOWER_SYSLOG_TCP_TLS_SECRET`

```

secretName: <secret>

```

String. The TLS secret name of the server for `syslogConfig`.

4. Optional: If applicable to your deployment, update the gateway CR file to configure `DataPowerService` APIs to customize your DataPower deployment. See [Customizing a DataPower deployment](#).
5. To enable mutual TLS between the Management client and the Gateway service's manager endpoint, add `mtlsValidateClient` to the `spec` section:

```

spec:
  mtlsValidateClient: true

```

This ensures that the gateway service authenticates incoming requests from the API Manager, such as gateway service registration, and publishing APIs to the gateway service. Specifically, the gateway service requires that incoming requests present a certificate that was signed by the same CA that was used to sign the gateway service management endpoint. The gateway service management endpoint secret is specified under `gatewayManagerEndpoint.hosts.secretName`. The API Manager's gateway client's TLS credentials are specified in the ManagementCluster CR under `gateway.client.secretName`.

Note: In releases previous to 10.0.5.3, this property is called `validateApimClient`.

6. Optional: If you want to enable JWT security for communications between the management and gateway, add and set the property `jwtksUrl`.

```

spec:
  ...
  jwtksUrl: <JWTKS URL>

```

where `<JWTKS URL>` is the URL of the JWKS endpoint hosted on the management subsystems. To find out the `jwtksUrl`, describe the management CR and check the `status`: section:

```

kubectl describe mgmt -n <namespace>
...
status:
  - name: jwtksUrl
    secretName: api-endpoint
    type: API
    uri: https://api.apic.acme.com/api/cloud/oauth2/certs

```

For more information on JWT security, see [Enable JWT security instead of mTLS](#).

Datapower version 10.5.0.4 only: To enable JWT on the gateway to analytics communications flow, you also must add a `dataPowerOverride` section to the gateway CR:

```

spec:
  jwtksUrl: <JWTKS URL>
  dataPowerOverride:
    image: customregistry.com/custom-image-datapower:10.5.0.5
    version: 10.5.0.5
    license: X-XXXX-XXXXXX

```

7. Optional: Update the gateway CR file to enable autoscaling of gateway pods to ensure high availability of DataPower pods. See [Enabling autoscaling of gateway pods](#).

8. Install the Gateway Custom Resource by applying the yaml file:
For a DataPower® API Gateway, run the following command:

```
oc apply -f apigateway_cr.yaml -n <namespace>
```

For a DataPower Gateway (v5 compatible), run the following command:

```
oc apply -f v5cgateway_cr.yaml -n <namespace>
```

9. To verify that the Gateway subsystem is fully installed, run the following command:

```
oc get GatewayCluster -n <namespace>
```

The installation is complete when the **READY** status is **True**, and the **SUMMARY** reports that all services are online (2/2) for all the Gateway subsystems that were installed. Example:

NAME	READY	SUMMARY	VERSION	RECONCILED	VERSION	AGE
gww6	True	2/2	<version>	<version-build>		7m32s

There is no need to wait for the **READY** status to be **True** before moving on to next Subsystem installation.

10. Verify that the gateway is ready to be registered with the Management subsystem by calling the **health** api on the `gatewayManagerEndpoint.hosts.name` defined in your yaml file: `https://gwd.$STACK_HOST/health`.

```
curl -k https://gwd.example.com/health  
{ "status": "ok" }
```

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the [Cloud Manager configuration checklist](#).

Previous topic: [Installing the Management subsystem in a shared namespace](#)

Next topic: [Installing the Portal subsystem in a shared namespace](#)

Installing the Portal subsystem in a shared namespace

Install the Portal subsystem by creating and applying the `portal_cr.yaml` file.

Before you begin

Complete the following tasks to prepare for deploying API Connect:

1. [Preparing for installation](#)
2. [Installing operators](#)
3. [Setting up a certificate issuer](#)

About this task

Use the OpenShift CLI to edit the custom resource template for the Portal subsystem, apply the resource, verify that the pods are up and running.

Procedure

1. Create a file that is called `portal_cr.yaml` and paste in these contents:

```
#  
# Licensed under the Apache License, Version 2.0 (the "License");  
# you may not use this file except in compliance with the License.  
# You may obtain a copy of the License at  
#  
# http://www.apache.org/licenses/LICENSE-2.0  
#  
# Unless required by applicable law or agreed to in writing, software  
# distributed under the License is distributed on an "AS IS" BASIS,  
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
# See the License for the specific language governing permissions and  
# limitations under the License.  
#  
  
apiVersion: portal.apiconnect.ibm.com/v1beta1  
kind: PortalCluster  
metadata:  
  name: portal  
  labels: {  
    app.kubernetes.io/instance: "portal",  
    app.kubernetes.io/managed-by: "ibm-apiconnect",  
    app.kubernetes.io/name: "portal"  
  }  
spec:
```

```

version: $APP_PRODUCT_VERSION
profile: $PROFILE
mgmtPlatformEndpointCASecret:
  secretName: ingress-ca
mgmtConsumerEndpointCASecret:
  secretName: ingress-ca
mgmtPlatformEndpointSvcCASecret:
  secretName: management-ca
mgmtConsumerEndpointSvcCASecret:
  secretName: management-ca
portalAdminEndpoint:
  annotations:
    cert-manager.io/issuer: ingress-issuer
  hosts:
  - name: api.portal.$STACK_HOST
    secretName: portal-admin
portalUIEndpoint:
  annotations:
    cert-manager.io/issuer: ingress-issuer
  hosts:
  - name: portal.$STACK_HOST
    secretName: portal-web
databaseVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 15Gi
databaseLogsVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 6Gi
webVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 8Gi
backupVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 15Gi
adminVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 6Gi
certVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 4Gi
adminClientSubjectDN: CN=portal-admin-client,O=cert-manager
microServiceSecurity: certManager
certManagerIssuer:
  name: selfsigning-issuer
  kind: Issuer
license:
  accept: $LICENSE_ACCEPTANCE
  use: $LICENSE_USE
  license: '$LICENSE_ID'

```

2. Edit the yaml file and set the variables:

```

$APP_PRODUCT_VERSION
API Connect application version for the subsystems.

version: <version_number>

Example version number: 10.0.5.4

```

\$PROFILE

Specify your portal subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see [Planning your deployment topology](#).

\$STACK_HOST

The desired ingress subdomain for the API Connect stack. Used when you specify endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can do one of the following:

- Subdomain customization only
Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **STACK_HOST** to be the desired ingress subdomain for the API Connect stack. For example, for a subdomain of **myhost.subnet.example.com**:

```

portalAdminEndpoint:
  < ... >
  hosts:
  - name: api.portal.myhost.subnet.example.com
    secret: portal-admin

portalUIEndpoint:
  < ... >
  hosts:
  - name: portal.myhost.subnet.example.com
    secret: portal-web

```

- Complete hostname customization
Change both the predefined prefixes and the **STACK_HOST** subdomain to match your desired hostnames.

For example, for **portalAdminEndpoint**, you can replace **api.portal.\$STACK_HOST** with **my.api.portal.myhost.subnet.example.com**, where **my.api.portal** replaces **api.portal**, and **myhost.subnet.example.com** replaces **STACK_HOST**:

```

portalAdminEndpoint:
  < ... >
  hosts:
  - name: my.api.portal.myhost.subnet.example.com
    secret: cm-endpoint

```

You can do this for some or all of the hostnames, depending on your customization requirements.

`$STORAGE_CLASS`

The Kubernetes storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: `oc get sc`. For example, for local storage:

```
databaseVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 120Gi
databaseLogsVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 12Gi
webVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 200Gi
backupVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 300Gi
adminVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 20Gi
certVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 4Gi
```

Note:

- The `databaseVolumeClaimTemplate` has a minimum value of 30Gi. The `webVolumeClaimTemplate` has a minimum value of 20Gi. The `backupVolumeClaimTemplate` has a minimum value of 30Gi. However, all three need to be sized according to the installation.
- The `adminVolumeClaimTemplate` and `databaseLogsVolumeClaimTemplate` in this example have their minimum values and you do not need to change them.
- The `certVolumeClaimTemplate` is used by the nginx pod and in 2dcdr (two data center disaster recovery) mode it is also used by the tunnel, db-remote and www-remote pods. It can be sized small, just like the `adminVolumeClaimTemplate`, and does not need to be increased based on the number of sites.

`$LICENSE_ACCEPTANCE`

Set `accept` to `true`. You must accept the license to successfully deploy API Connect.

`$LICENSE_USE`

Set `use` to either `production` or `nonproduction` to match the license that you purchased.

`$LICENSE_ID`

Set `license` to the license ID for the version of API Connect that you purchased. See [API Connect licenses](#).

3. Optional: If you want to disable mTLS for communications between the management and portal, and enable JWT instead, then add and set the properties `mtlsValidateClient` and `jwtksUrl`.

```
spec:
  ...
  mtlsValidateClient: false
  jwtksUrl: <JWTKS URL>
```

where `<JWTKS URL>` is the URL of the JWKS endpoint hosted on the management subsystems. To find out the `jwtksUrl`, describe the management CR and check the `status` section:

```
kubectl describe mgmt -n <namespace>
...
status:
  - name: jwtksUrl
    secretName: api-endpoint
    type: API
    uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information on JWT security, see [Enable JWT security instead of mTLS](#).

4. Install the Portal subsystem by applying the modified CR with the following command:

```
oc apply -f portal_cr.yaml -n <namespace>
```

5. Verify that the Portal subsystem is fully installed:

```
oc get PortalCluster -n <namespace>
```

The installation is complete when `STATUS` reports `Running`, `Ready` reports that all services are online (3/3), and `MESSAGE` reports "Ready for API Cloud Manager service registration", for example:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	MESSAGE	AGE
portal	3/3	Running	<version>	<version>		Ready for API Cloud Manager service registration	2h

You do not need to wait for the portal installation to complete before you move on to the next subsystem installation.

6. Verify that you can access the portal web endpoint by browsing to the `portalUIEndpoint.hosts.name` as configured in your yaml file:

```
curl -k https://portal.$STACK_HOST
<!DOCTYPE html>
<html>
<head>
<title>Developer Portal is ready to create sites!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>Developer Portal is ready to create sites!</h1>
<p>If you see this page, the web server is successfully installed and
```

```
working. Further configuration is required.</p>
</body>
</html>
```

7. It is highly recommend that you retain the Portal subsystem encryption secret in a safe location, in case you need to restore the Portal subsystem during a disaster recovery scenario.

The encryption secret is created when the Portal subsystem is initially deployed. You can obtain the name of the encryption secret from the Portal subsystem custom resource after the installation is complete:

a. Obtain the name of the encryptionSecret:

```
oc get ptl portal -o yaml -n <namespace> | grep encryptionSecret
encryptionSecret: portal-enc-key
```

The name of encryption secret is `portal-enc-key`.

b. Save the secret yaml in a safe location:

```
oc get secret portal-enc-key -o yaml -n <namespace> > portal-enc-key.yaml
```

8. Backup the Portal installation. See [Backing up and restoring Developer Portal on OpenShift and Cloud Pak for Integration](#).

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the [Cloud Manager configuration checklist](#).

Previous topic: [Installing the Gateway subsystem in a shared namespace](#)

Next topic: [Installing the Analytics subsystem in a shared namespace](#)

Installing the Analytics subsystem in a shared namespace

Install the analytics subsystem by creating and applying the `analytics_cr.yaml` file.

Before you begin

Complete the following tasks to prepare for deploying API Connect:

1. [Preparing for installation](#)
2. [Installing operators](#)
3. [Setting up a certificate issuer](#)

About this task

Use the OpenShift CLI to edit the custom resource template for the analytics subsystem, apply the resource, verify that the pods are up and running.

Procedure

1. Create a file that is called `analytics_cr.yaml` and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsCluster
metadata:
  name: analytics
  labels: {
    app.kubernetes.io/instance: "analytics",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "analytics"
  }
spec:
  version: $APP_PRODUCT_VERSION
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
  profile: $PROFILE
  microServiceSecurity: certManager
  certManagerIssuer:
```

```

name: selfsigning-issuer
kind: Issuer
ingestion:
  endpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: ai.$STACK_HOST
        secretName: analytics-ai-endpoint
    clientSubjectDN: CN=analytics-ingestion-client,O=cert-manager
storage:
  type: shared # change to 'dedicated' if you want separate master and storage pods. Three-replica deployments only.
  shared:
    volumeClaimTemplate:
      storageClassName: $STORAGE_CLASS
      volumeSize: $DATA_VOLUME_SIZE
  # master: # uncomment this section if you set storage.type = dedicated.
  # volumeClaimTemplate:
  #   storageClassName: $STORAGE_CLASS

```

2. Edit the yaml file and set the variables:

\$APP_PRODUCT_VERSION

API Connect application version for the subsystems.

```
version: <version_number>
```

Example version number: 10.0.5.4

\$PROFILE

Specify your analytics subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see [Planning your deployment topology](#).

\$STACK_HOST

The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can customize the subdomain or the complete hostname:

- Subdomain customization only
Accept the prefixes predefined for the ingress hostnames to use and replace all instances of `$STACK_HOST` with the desired ingress subdomain for the API Connect stack. For example, if your host is `myhost.subnet.example.com`:

```

ingestion:
  endpoint:
    < ... >
  hosts:
    - name: ai.myhost.subnet.example.com
      secret: analytics-ai-endpoint
    < ... >

```

- Complete hostname customization
Change both the predefined prefixes and the `$STACK_HOST` subdomain to match your desired hostnames.

For example, you can replace `$STACK_HOST` with `my.analytics.ingestion.myhost.subnet.example.com`, where `my.analytics.ingestion` replaces `ai`, and `myhost.subnet.example.com` replaces `STACK_HOST`.

```

ingestion:
  endpoint:
    < ... >
  hosts:
    - name: my.analytics.ingestion.myhost.subnet.example.com
      secret: analytics-ai-endpoint
    < ... >

```

\$STORAGE_CLASS

The Kubernetes storage class to be used for persistent volume claims (for more information, see [Planning the analytics profile, storage class, and storage type](#)). Find the available storage classes in the target cluster by running the following command: `oc get sc`. Example:

```

storage:
  type: shared
  shared:
    volumeClaimTemplate:
      storageClassName: ceph-block

```

\$DATA_VOLUME_SIZE

Size of storage allocated for data. To estimate the storage space you require, see [Estimating internal storage space](#). If `$DATA_VOLUME_SIZE` is not specified, it defaults to 50Gi. See [Reference: Fields in the Analytics CR](#). Example:

```

storage:
  type: shared
  shared:
    volumeClaimTemplate:
      storageClassName: ceph-block
      volumeSize: 100Gi

```

\$LICENSE_ACCEPTANCE

Set `accept` to `true`. You must accept the license to successfully deploy API Connect.

\$LICENSE_USE

Set `use` to either `production` or `nonproduction` to match the license that you purchased.

\$LICENSE_ID

Set `license` to the license ID for the version of API Connect that you purchased. See [API Connect licenses](#).

3. Optional: Three replica deployments only: Enable dedicated storage.

If you want to use dedicated storage, update the `spec.storage` section of the yaml file as follows:

```
storage:
  type: dedicated
  shared:
    volumeClaimTemplate:
      storageClassName: $STORAGE_CLASS
      volumeSize: $DATA_VOLUME_SIZE
  master:
    volumeClaimTemplate:
      storageClassName: $STORAGE_CLASS
```

For more information about dedicated storage, see [Dedicated storage instead of shared storage](#).

4. Optional: If you want to disable mTLS for communications between the management and analytics subsystem, and between the gateway and analytics subsystem, and enable JWT instead, then add and set the properties `mtlsValidateClient` and `jwtksUrl`.

```
spec:
  ...
  mtlsValidateClient: false
  jwtksUrl: <JWTKS_URL>
```

where `<JWTKS URL>` is the URL of the JWKS endpoint hosted on the management subsystems. To find out the `jwtksUrl`, describe the management CR and check the `status` section:

```
kubectl describe mgmt -n <namespace>
...
status:
- name: jwtksUrl
  secretName: api-endpoint
  type: API
  uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information on JWT security, see [Enable JWT security instead of mTLS](#).

Note: It is not possible to use JWT on the V5 compatible gateway to analytics message flow.

5. If you want to enable additional analytics features, you can set them in the `analytics_cr.yaml` file as described here: [Analytics subsystem CR configured settings](#)

6. Install the analytics subsystem by applying the modified CR with the following command:

```
oc apply -f analytics_cr.yaml -n <namespace>
```

7. Verify that the analytics subsystem is fully installed:

```
oc get AnalyticsCluster -n <namespace>
```

The installation is complete when `READY` shows all pods running (n/n), and the `STATUS` reports `Running`. Example:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
analytics	n/n	Running	10.0.5.3	10.0.5.3-1281	86m	

It is not necessary to wait for analytics installation to complete before you move on to the next subsystem installation.

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the [Cloud Manager configuration checklist](#).

Previous topic: [Installing the Portal subsystem in a shared namespace](#)

Installing with subsystem CRs in different namespaces or environments on OpenShift

Install API Connect subsystems in different namespaces, for example Management subsystem in a namespace that is called `apicmgmt`, Gateway in a namespace called `apicgateway`. These same steps can also be used for installing subsystems in different environments.

About this task

Install API Connect subsystems where you have separate namespaces for each subsystem.

1. [Installing operators](#)

Install operators for API Connect, DataPower, and common services so that you can deploy API Connect subsystems across multiple namespaces on OpenShift. If deploying subsystems in different environments then the corresponding operator will need to be installed in each environment, for example the DataPower operator in your Gateway environment, and the API Connect operator in your Portal environment.

2. [Installing the Management subsystem](#)

Install the Management subsystem by creating and applying the `ingress-issuer` and `mgmt_cr` yaml files.

3. [Extracting the Management ingress-ca certificates](#)

In order for the Management subsystem to be able to communicate with subsystems in different namespaces, these subsystems need to have the same `ingress-ca` certificates as the Management subsystem.

4. [Installing the Gateway subsystem](#)

Install the Gateway subsystem by creating and applying the `ingress-ca`, `common-issuer`, `gateway-certs` and `gateway_cr` yaml files.

5. [Installing the Portal subsystem](#)

Install the Portal subsystem by creating and applying the ingress-ca, common-issuer, and portal_cr yaml files.

6. [Installing the Analytics subsystem](#)

Install the Analytics subsystem by creating and applying the ingress-ca, common-issuer, and analytics_cr yaml files.

Installing operators

Install operators for API Connect, DataPower, and common services so that you can deploy API Connect subsystems across multiple namespaces on OpenShift. If deploying subsystems in different environments then the corresponding operator will need to be installed in each environment, for example the DataPower operator in your Gateway environment, and the API Connect operator in your Portal environment.

Before you begin

The API Connect operator and operand must be from the same release and fix pack level. Table 1 lists the current version of the operator and operand for API Connect.

Table 1. API Connect operator and operand versions

API Connect (operand)	Operator channel version	Highest operator version
10.0.5.4	v3.4	3.4.0

You can use any combination of API Connect 10.0.5.x with DataPower Gateway 10.5.0.x or DataPower API Gateway 10.5.0.x.

Procedure

1. Obtain an entitlement key for the Entitled Registry:
 - a. Log in to the [IBM Container Library](#).
 - b. In the Container software library, select Get entitlement key.
 - c. After the Access your container software heading, click Copy key.
 - d. Copy the key to a safe location.
2. Create a global pull secret:
 - a. Open the OpenShift web console, click Workloads > Secrets.
 - b. Select the project `openshift-config`.
 - c. Select `pull-secret` from the secret list.
 - d. Select Actions > Edit Secret > Add Credentials.
 - e. Set Registry Server Address to `cp.icr.io`.
 - f. Set Username to `cp`.
 - g. Set Password to the entitlement key generated in Step 1.
 - h. Click Save.
3. Add the IBM Operator Catalog source:
 - a. In the OpenShift web console, and click + (in the page banner) to open the Import YAML dialog box.
 - b. Paste the following resource definition into the dialog box:

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: ibm-operator-catalog
  namespace: openshift-marketplace
spec:
  displayName: "IBM Operator Catalog"
  publisher: IBM
  sourceType: grpc
  image: icr.io/cpopen/ibm-operator-catalog
  updateStrategy:
    registryPoll:
      interval: 45m
```

- c. Click Create.
4. Create the `apiconnect` subscription.
 - a. Create the `apiconnect` subscription with the appropriate channel by creating a file called `apic-sub.yaml` and pasting in the following contents:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-apiconnect
  namespace: openshift-operators
spec:
  channel: v3.4
  name: ibm-apiconnect
  source: ibm-operator-catalog
  sourceNamespace: openshift-marketplace
```

- b. Apply the subscription with the following command:

```
oc apply -f apic-sub.yaml
```

- c. Select Operators > Installed Operators, and ensure that Project: All Projects is selected.
If any operators such as `ibm-apiconnect` or `ibm-cert-manager-operator` show the status of "Upgrade available", approve the upgrade by completing the following steps:
 - i. Click Upgrade available.
 - ii. Click Preview InstallPlan.
 - iii. Click Approve.
 - iv. Wait for the **IBM API Connect** and **IBM DataPower Gateway** operators to install.

The **IBM DataPower Gateway** operator is a prerequisite to API Connect and must not be removed.

5. (10.0.5.3 and later) Create the **ibm-common-services-operator** subscription, if it does not already exist. This step only applies when you install version 10.0.5.3 or later. If you are installing an older version of API Connect, skip this step.

- a. Create a file called `common-services-sub.yaml` and paste in the following contents:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-common-service-operator
  namespace: openshift-operators
spec:
  channel: <channel>
  name: ibm-common-service-operator
  source: ibm-operator-catalog
  sourceNamespace: openshift-marketplace
```

Where `<channel>` is one of the following values:

- `v3.23` if you are using IBM Cloud Pak foundational services for CD (continuous delivery)
- `v3` if you are using IBM Cloud Pak foundational services for LTSR (long term service release)

- b. Apply the subscription with the following command:

```
oc apply -f common-services-sub.yaml
```

- c. Select **Operators**, **Installed Operators**, and ensure that **Project: All Projects** is selected.

If any operators such as **ibm-apiconnect** or **ibm-cert-manager-operator** show the status of "Upgrade available", approve the upgrade by completing the following steps:

- Click Upgrade available.
- Click Preview InstallPlan.
- Click Approve.
- Check the **ibm-common-services** namespace and ensure that all operators with a status of "Upgrade available" are approved.
- Wait for the **IBM Cloud Pak foundational services, IBM NamespaceScope**, and **Operand Deployment Lifecycle Manager** operators to install.

6. Create the namespaces for your API Connect subsystems either by selecting **Project**, **Create Project** in the UI, or with the command:

```
oc create ns <APIC-namespace>
```

Note: The following namespaces cannot be used to install API Connect subsystems because Red Hat OpenShift Container Platform (OCP) restricts the use of default namespaces for installing non-cluster services:

- default
- kube-system
- kube-public
- openshift-node
- openshift-infra
- openshift

7. (10.0.5.3 and later) Install cert manager:

This step only applies when you install version 10.0.5.3 or later. If you are installing an older version of API Connect, skip this step.

- a. Create a file called `cert-manager-operand-request.yaml` and paste in the following content:

```
apiVersion: operator.ibm.com/v1alpha1
kind: OperandRequest
metadata:
  name: ibm-apiconnect-cert-manager
  namespace: <namespace>
spec:
  requests:
  - operands:
    - name: ibm-cert-manager-operator
      registry: common-service
      registryNamespace: ibm-common-services
```

- b. Create the **operandRequest** for cert-manager by running the following command:

```
oc apply -f cert-manager-operand-request.yaml
```

- c. Select **Operators**, **Installed Operators**, and ensure that **Project: All Projects** is selected.

Wait for the **ibm-cert-manager-operator** to display and if it shows the status of "Upgrade available", approve the upgrade by completing the following steps:

- Click Upgrade available.
- Click Preview InstallPlan.
- Click Approve.
- Wait for the **IBM Cert Manager** operator to install.

What to do next

Proceed to install the API Connect subsystems as explained in : [Installing the Management subsystem](#)

Next topic: [Installing the Management subsystem](#)

Installing the Management subsystem

Install the Management subsystem by creating and applying the ingress-issuer and mgmt_cr yaml files.

Before you begin

When installing subsystems in different namespaces you must first create the cert-manager issuers and certificates required by that subsystem in the same namespace.

1. Create a file called `ingress-issuer-management.yaml` and paste in the following:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
---

apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "selfsigning-issuer"
  }
spec:
  selfSigned: {}
---

apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-issuer"
  }
spec:
  ca:
    secretName: ingress-ca
---

apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ingress-ca
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-ca"
  }
spec:
  secretName: ingress-ca
  commonName: "ingress-ca"
  usages:
  - digital signature
  - key encipherment
  - cert sign
  isCA: true
  duration: 87600h # 10 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  issuerRef:
    name: selfsigning-issuer
    kind: Issuer
---

apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: portal-admin-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "portal-admin-client"
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: portal-admin-client
  secretName: portal-admin-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
```

```

- "key encipherment"
duration: 17520h # 2 years
renewBefore: 720h # 30 days
privateKey:
  rotationPolicy: Always
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-client-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-client-client"
  }
spec:
  subject:
    organizations:
      - cert-manager
  commonName: gateway-client-client
  secretName: gateway-client-client
  issuerRef:
    name: ingress-issuer
  usages:
    - "client auth"
    - "signing"
    - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: analytics-ingestion-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "analytics-ingestion-client"
  }
spec:
  subject:
    organizations:
      - cert-manager
  commonName: analytics-ingestion-client
  secretName: analytics-ingestion-client
  issuerRef:
    name: ingress-issuer
  usages:
    - "client auth"
    - "signing"
    - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always

```

2. Apply the above yaml file with: `oc apply -f ingress-issuer-management.yaml -n <management namespace>`
3. Confirm that the issuers were created and are in ready state:

```

oc get issuers -n <management namespace>
NAME                READY
ingress-issuer      True
selfsigning-issuer True

```

Procedure

1. Create a file that is called `mgmt_cr.yaml` and paste in these contents:

```

#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#   http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementCluster
metadata:
  name: management
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",

```

```

    app.kubernetes.io/name: "management"
  }
spec:
  version: $APP_PRODUCT_VERSION
  profile: $PROFILE
  portal:
    admin:
      secretName: portal-admin-client
  analytics:
    ingestion:
      secretName: analytics-ingestion-client
  gateway:
    client:
      secretName: gateway-client-client
  cloudManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: admin.$STACK_HOST
        secretName: cm-endpoint
  apiManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: manager.$STACK_HOST
        secretName: apim-endpoint
  platformAPIEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: api.$STACK_HOST
        secretName: api-endpoint
  consumerAPIEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: consumer.$STACK_HOST
        secretName: consumer-endpoint
  databaseVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
  microServiceSecurity: certManager
  certManagerIssuer:
    name: selfsigning-issuer
    kind: Issuer
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'

```

2. Edit the yaml file and replace the variables:

\$APP_PRODUCT_VERSION

API Connect application version for the subsystems.

```
version: <version_number>
```

Example version number: 10.0.5.4

\$PROFILE

Specify your Management subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see [Planning your deployment topology](#).

\$STACK_HOST

The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore " " character. You can do one of the following:

- Subdomain customization only

Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **STACK_HOST** to be the desired ingress subdomain for the API Connect stack. For example, if your host is **myhost.subnet.example.com**:

```

cloudManagerEndpoint:
  < ... >
  hosts:
    - name: admin.myhost.subnet.example.com
      secret: cm-endpoint

apiManagerEndpoint:
  < ... >
  hosts:
    - name: manager.myhost.subnet.example.com
      secret: apim-endpoint

platformAPIEndpoint:
  < ... >
  hosts:
    - name: api.myhost.subnet.example.com
      secret: api-endpoint

consumerAPIEndpoint:
  < ... >
  hosts:
    - name: consumer.myhost.subnet.example.com
      secret: consumer-endpoint

```

- Complete hostname customization

Change both the predefined prefixes and the `STACK_HOST` subdomain to match your desired hostnames.

For example, for `cloudManagerEndpoint`, you can replace `admin.$STACK_HOST` with `my.cloudmgr.myhost.subnet.example.com`, where `my.cloudmgr` replaces `admin`, and `myhost.subnet.example.com` replaces `STACK_HOST`. For example:

```
cloudManagerEndpoint:
  < ... >
  hosts:
  - name: my.cloudmgr.myhost.subnet.example.com
    secret: cm-endpoint
```

You can do this for some or all of the hostnames, depending on your customization requirements.

`$STORAGE_CLASS`

The OCP storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: `oc get sc`.

```
storageClassName: local-storage
```

`$LICENSE_ACCEPTANCE`

Set `accept` to `true`. You must accept the license to successfully deploy API Connect.

`$LICENSE_USE`

Set `use` to either `production` or `nonproduction` to match the license that you purchased.

`$LICENSE_ID`

Set `license`: to the license ID for the version of API Connect that you purchased. See [API Connect licenses](#).

3. Configure database backups for the Management subsystem.

To successfully restore API Connect in the event of a disaster where you must redeploy a new cluster, you must configure the database backup settings before the initial installation of the Management subsystem. For instructions on configuring backup settings, see [Configuring backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration](#).

4. Install the Management subsystem by applying the modified CR with the following command:

```
oc apply -f mgmt_cr.yaml -n <management_namespace>
```

5. Verify that the Management subsystem is fully installed by running the following command:

```
oc get ManagementCluster -n <management_namespace>
```

The installation is complete when the `READY` status is `True`, and the `SUMMARY` reports that all services are online:

NAME	READY	SUMMARY	VERSION	RECONCILED	VERSION	AGE
management	True	16/16	<version>		<version-build>	7m17s

6. Verify that you can log in to the API Connect Cloud Manager UI:

- a. Determine the URL for the Cloud Manager UI by running the following command to view the API Connect endpoints:

```
oc get routes -n <management_namespace>
```

- b. Locate the `management-admin` endpoint, and note it down.

- c. Determine the Cloud Manager administrator password by running the following commands:

```
oc get secret -n <management-namespace> | grep management-admin-secret
```

```
oc get secret -n <management-namespace> <secret_name_from_previous_command> -o jsonpath="{.data.password}" | base64 -d && echo ""
```

- d. Open a browser and go to the `management-admin-apic` endpoint.

- e. Log in to the Cloud Manager as `admin` with the administrator password.

7. Store all secrets and passwords in a safe place.

8. Prepare the Management subsystem for disaster recovery.

Follow the appropriate instructions for your environment:

- [Preparing the Management subsystem for disaster recovery on OpenShift](#)
- [Preparing the Management subsystem for disaster recovery on Cloud Pak for Integration](#)

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the [Cloud Manager configuration checklist](#).

Previous topic: [Installing operators](#)

Next topic: [Extracting the Management ingress-ca certificates](#)

Extracting the Management ingress-ca certificates

In order for the Management subsystem to be able to communicate with subsystems in different namespaces, these subsystems need to have the same ingress-ca certificates as the Management subsystem.

About this task

This task describes how to extract the ingress-ca certificates to a yaml file, ready for applying to the subsystems that are in different namespaces.

Procedure

1. Export the ingress-ca secret to a yaml file with: `oc -n <management namespace> get secret ingress-ca -o yaml > ingress-ca.yaml`
2. Edit the ingress-ca.yaml file and remove the following properties: metadata.creationTimestamp, metadata.namespace, metadata.resourceVersion, metadata.uid, metadata.selfLink
3. Keep this file safe. You will need to apply it in the namespaces of your other subsystems so that they can communicate with the Management subsystem.

Previous topic: [Installing the Management subsystem](#)

Next topic: [Installing the Gateway subsystem](#)

Installing the Gateway subsystem

Install the Gateway subsystem by creating and applying the ingress-ca, common-issuer, gateway-certs and gateway_cr yaml files.

Before you begin

When installing subsystems in different namespaces you must first create the cert-manager issuers and certificates required by that subsystem in the same namespace.

1. Using the file obtained from [Extracting the Management ingress-ca certificates](#), run: `oc apply -f ingress-ca.yaml -n <gateway namespace>`
2. Create a file called `common-issuer-and-gateway-certs.yaml` and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "selfsigning-issuer"
  }
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-issuer"
  }
spec:
  ca:
    secretName: ingress-ca
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-service
  labels: {
    app.kubernetes.io/instance: "gatewaycluster",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-service"
  }
spec:
  commonName: gateway-service
  secretName: gateway-service
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
```

```

---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-peering
  labels: {
    app.kubernetes.io/instance: "gatewaycluster",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-peering"
  }
spec:
  commonName: gateway-peering
  secretName: gateway-peering
  issuerRef:
    name: ingress-issuer
  usages:
  - "server auth"
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always

```

3. Apply the above yaml file with: `oc apply -f common-issuer-and-gateway-certs.yaml -n <gateway namespace>`
4. Confirm that the issuers were created and are in ready state:

```

oc get issuers -n <gateway namespace>
NAME                READY
ingress-issuer      True
selfsigning-issuer True

```

5. Confirm that the gateway secrets were created with `oc get secrets -n <gateway namespace>`:

```

oc get secrets -n <gateway namespace>
NAME                TYPE                DATA  AGE
...
gateway-peering    kubernetes.io/tls   3      2m
gateway-service    kubernetes.io/tls   3      2m3s
...

```

About this task

Use the OpenShift CLI to edit the custom resource template for the Gateway, apply the resource, verify that the pods are up and running.

Procedure

1. Create the gateway admin secret with: `oc -n <gateway namespace> create secret generic admin-secret --from-literal=password=admin`
2. If you are installing the API Gateway (v6), create a yaml file that is called `apigateway_cr.yaml` and paste in these contents:

```

#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

apiVersion: gateway.apiconnect.ibm.com/v1beta1
kind: GatewayCluster
metadata:
  name: gwv6
  labels: {
    app.kubernetes.io/instance: "gateway",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gwv6"
  }
spec:
  version: $APP_PRODUCT_VERSION
  profile: $PROFILE
  apicGatewayServiceV5CompatibilityMode: false
  mgmtPlatformEndpointCASecret:
    secretName: ingress-ca
  mgmtPlatformEndpointSvcCASecret:
    secretName: management-ca
  gatewayEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: rgw.$STACK_HOST
        secretName: gwv6-endpoint

```

```

gatewayManagerEndpoint:
  annotations:
    cert-manager.io/issuer: ingress-issuer
  hosts:
    - name: rgwd.$STACK_HOST
      secretName: gwv6-manager-endpoint
apicGatewayServiceTLS:
  secretName: gateway-service
apicGatewayPeeringTLS:
  secretName: gateway-peering
datapowerLogLevel: 3
license:
  accept: $LICENSE_ACCEPTANCE
  use: $LICENSE_USE
  license: '$LICENSE_ID'
tokenManagementService:
  enabled: true
  storage:
    storageClassName: $STORAGE_CLASS
    volumeSize: 30Gi
adminUser:
  secretName: admin-secret
# syslogConfig:
#   enabled: false # if true, provide below details
#   remoteHost: $DATAPOWER_SYSLOG_TCP_REMOTE_HOST # must be a string
#   remotePort: $DATAPOWER_SYSLOG_TCP_REMOTE_PORT # must be an int
#   secretName: $DATAPOWER_SYSLOG_TCP_TLS_SECRET # must be a string

```

If you are installing the v5 Gateway, create a yaml file that is called v5gateway_cr.yaml and paste in these contents:

```

#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#   http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

apiVersion: gateway.apiconnect.ibm.com/v1beta1
kind: GatewayCluster
metadata:
  name: gwv5
  labels: {
    app.kubernetes.io/instance: "gateway",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gwv5"
  }
spec:
  version: $APP_PRODUCT_VERSION
  profile: $PROFILE
  apicGatewayServiceV5CompatibilityMode: true
  gatewayEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: gw.$STACK_HOST
        secretName: gwv5-endpoint
  gatewayManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: gwd.$STACK_HOST
        secretName: gwv5-manager-endpoint
  apicGatewayServiceTLS:
    secretName: gateway-service
  apicGatewayPeeringTLS:
    secretName: gateway-peering
  datapowerLogLevel: 3
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
  adminUser:
    secretName: admin-secret
# syslogConfig:
#   enabled: false # if true, provide below details
#   remoteHost: $DATAPOWER_SYSLOG_TCP_REMOTE_HOST # must be a string
#   remotePort: $DATAPOWER_SYSLOG_TCP_REMOTE_PORT # must be an int
#   secretName: $DATAPOWER_SYSLOG_TCP_TLS_SECRET # must be a string

```

3. Edit the yaml file and set the variables:

```

$APP_PRODUCT_VERSION
API Connect application version for the subsystems.

```

```

version: <version_number>

```

Example version number: 10.0.5.4

\$PROFILE

Specify your Gateway subsystem profile, where **n** indicates number of replicas, **e** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see [Planning your deployment topology](#).

\$STACK_HOST

The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can do one of the following:

- Subdomain customization only
Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **STACK_HOST** to be the desired ingress subdomain for the API Connect stack. For example, if your host is `myhost.subnet.example.com`:

```
gatewayEndpoint:
  < ... >
  hosts:
  - name: rgw.myhost.subnet.example.com
    secret: gwv6-endpoint

gatewayManagerEndpoint:
  < ... >
  hosts:
  - name: rgwd.myhost.subnet.example.com
    secret: gwv6-manager-endpoint
```

- Complete hostname customization
Change both the predefined prefixes and the **STACK_HOST** subdomain to match your desired hostnames.

For example, for `gatewayEndpoint`, you can replace `rgw.$STACK_HOST` with `my.gateway.endpoint.myhost.subnet.example.com`, where `my.gateway.endpoint` replaces `rgw`, and `myhost.subnet.example.com` replaces `$STACK_HOST`:

```
gatewayEndpoint:
  < ... >
  hosts:
  - name: my.gateway.endpoint.myhost.subnet.example.com
    secret: gwv6-endpoint
```

You can do this for some or all of the hostnames, depending on your customization requirements.

\$STORAGE_CLASS

The OCP storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: `oc`

```
get
sc.
```

```
storageClassName: local-storage
```

\$LICENSE_ACCEPTANCE

Set `accept` to `true`. You must accept the license to successfully deploy API Connect.

\$LICENSE_USE

Set `use` to either `production` or `nonproduction` to match the license that you purchased.

\$LICENSE_ID

Set `license`: to the license ID for the version of API Connect that you purchased. See [API Connect licenses](#).

syslogConfig

To enable this feature uncomment this section, set `enabled`: `true`, and set the properties:

\$DATAPOWER_SYSLOG_TCP_REMOTE_HOST

```
remoteHost: <remote.host>
```

String. The hostname of the server for `syslogConfig`.

\$DATAPOWER_SYSLOG_TCP_REMOTE_PORT

```
remotePort: <remote.port.number>
```

Integer. The port number of the server for `syslogConfig`.

\$DATAPOWER_SYSLOG_TCP_TLS_SECRET

```
secretName: <secret>
```

String. The TLS secret name of the server for `syslogConfig`.

4. Optional: If applicable to your deployment, update the gateway CR file to configure `DataPowerService` APIs to customize your DataPower deployment. See [Customizing a DataPower deployment](#).
5. To enable mutual TLS between the Management client and the Gateway service's manager endpoint, add `mtlsValidateClient` to the `spec` section:

```
spec:
  mtlsValidateClient: true
```

This ensures that the gateway service authenticates incoming requests from the API Manager, such as gateway service registration, and publishing APIs to the gateway service. Specifically, the gateway service requires that incoming requests present a certificate that was signed by the same CA that was used to sign the gateway service management endpoint. The gateway service management endpoint secret is specified under `gatewayManagerEndpoint.hosts.secretName`. The API Manager's gateway client's TLS credentials are specified in the ManagementCluster CR under `gateway.client.secretName`.

Note: In releases previous to 10.0.5.3, this property is called `validateApimClient`.

6. Optional: If you want to enable JWT security for communications between the management and gateway, add and set the property `jwtksUrl`.

```
spec:
  ...
  jwtksUrl: <JWTKS URL>
```

where `<JWTKS URL>` is the URL of the JWKS endpoint hosted on the management subsystems. To find out the `jwtksUrl`, describe the management CR and check the `status`: section:

```
kubectl describe mgmt -n <namespace>
...
status:
```

```
- name: jwksUrl
  secretName: api-endpoint
  type: API
  uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information on JWT security, see [Enable JWT security instead of mTLS](#).

DataPower version 10.5.0.4 only: To enable JWT on the gateway to analytics communications flow, you also must add a `dataPowerOverride` section to the gateway CR:

```
spec:
  jwksUrl: <JWKS URL>
  dataPowerOverride:
    image: customregistry.com/custom-image-datapower:10.5.0.5
    version: 10.5.0.5
    license: X-XXXX-XXXXXX
```

7. Optional: Update the gateway CR file to enable autoscaling of gateway pods to ensure high availability of DataPower pods. See [Enabling autoscaling of gateway pods](#).

8. Install the Gateway Custom Resource by applying the yaml file:
For a DataPower® API Gateway, run the following command:

```
oc apply -f apigateway_cr.yaml -n <namespace>
```

For a DataPower Gateway (v5 compatible), run the following command:

```
oc apply -f v5cgateway_cr.yaml -n <namespace>
```

9. To verify that the Gateway subsystem is fully installed, run the following command:

```
oc get GatewayCluster -n <namespace>
```

The installation is complete when the **READY** status is **True**, and the **SUMMARY** reports that all services are online (2/2) for all the Gateway subsystems that were installed. Example:

NAME	READY	SUMMARY	VERSION	RECONCILED	VERSION	AGE
gww6	True	2/2	<version>	<version-build>		7m32s

There is no need to wait for the **READY** status to be **True** before moving on to next Subsystem installation.

10. Verify that the gateway is ready to be registered with the Management subsystem by calling the **health** api on the `gatewayManagerEndpoint.hosts.name` defined in your yaml file: `https://gwd.$STACK_HOST/health`.

```
curl -k https://gwd.example.com/health
{"status": "ok"}
```

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the [Cloud Manager configuration checklist](#).

Previous topic: [Extracting the Management ingress-ca certificates](#)

Next topic: [Installing the Portal subsystem](#)

Installing the Portal subsystem

Install the Portal subsystem by creating and applying the ingress-ca, common-issuer, and portal_cr yaml files.

Before you begin

When installing subsystems in different namespaces you must first create the cert-manager issuers and certificates required by that subsystem in the same namespace.

- Using the file obtained from [Extracting the Management ingress-ca certificates](#), run: `oc apply -f ingress-ca.yaml -n <portal namespace>`
- Create a file called `common-issuer.yaml` and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
```

```

labels: {
  app.kubernetes.io/instance: "management",
  app.kubernetes.io/managed-by: "ibm-apiconnect",
  app.kubernetes.io/name: "selfsigning-issuer"
}
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-issuer"
  }
spec:
  ca:
    secretName: ingress-ca

```

3. Apply the above yaml file with: `oc apply -f common-issuer.yaml -n <portal namespace>`
4. Confirm that the issuers were created and are in ready state:

```

oc get issuers -n <portal namespace>
NAME                READY
ingress-issuer      True
selfsigning-issuer  True

```

About this task

Use the OpenShift CLI to edit the custom resource template for the Portal subsystem, apply the resource, verify that the pods are up and running.

Procedure

1. Create a file that is called `portal_cr.yaml` and paste in these contents:

```

#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalCluster
metadata:
  name: portal
  labels: {
    app.kubernetes.io/instance: "portal",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "portal"
  }
spec:
  version: $APP_PRODUCT_VERSION
  profile: $PROFILE
  mgmtPlatformEndpointCASecret:
    secretName: ingress-ca
  mgmtConsumerEndpointCASecret:
    secretName: ingress-ca
  mgmtPlatformEndpointSvcCASecret:
    secretName: management-ca
  mgmtConsumerEndpointSvcCASecret:
    secretName: management-ca
  portalAdminEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: api.portal.$STACK_HOST
        secretName: portal-admin
  portalUIEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: portal.$STACK_HOST
        secretName: portal-web
  databaseVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 15Gi
  databaseLogsVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 6Gi

```

```

webVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 8Gi
backupVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 15Gi
adminVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 6Gi
certVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 4Gi
adminClientSubjectDN: CN=portal-admin-client,O=cert-manager
microServiceSecurity: certManager
certManagerIssuer:
  name: selfsigning-issuer
  kind: Issuer
license:
  accept: $LICENSE_ACCEPTANCE
  use: $LICENSE_USE
  license: '$LICENSE_ID'

```

2. Edit the yaml file and set the variables:

\$APP_PRODUCT_VERSION

API Connect application version for the subsystems.

```
version: <version_number>
```

Example version number: 10.0.5.4

\$PROFILE

Specify your portal subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see [Planning your deployment topology](#).

\$STACK_HOST

The desired ingress subdomain for the API Connect stack. Used when you specify endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can do one of the following:

- Subdomain customization only

Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **STACK_HOST** to be the desired ingress subdomain for the API Connect stack. For example, for a subdomain of **myhost.subnet.example.com**:

```

portalAdminEndpoint:
  < ... >
  hosts:
    - name: api.portal.myhost.subnet.example.com
      secret: portal-admin

portalUIEndpoint:
  < ... >
  hosts:
    - name: portal.myhost.subnet.example.com
      secret: portal-web

```

- Complete hostname customization

Change both the predefined prefixes and the **\$STACK_HOST** subdomain to match your desired hostnames.

For example, for **portalAdminEndpoint**, you can replace **api.portal.\$STACK_HOST** with **my.api.portal.myhost.subnet.example.com**, where **my.api.portal** replaces **api.portal**, and **myhost.subnet.example.com** replaces **\$STACK_HOST**:

```

portalAdminEndpoint:
  < ... >
  hosts:
    - name: my.api.portal.myhost.subnet.example.com
      secret: cm-endpoint

```

You can do this for some or all of the hostnames, depending on your customization requirements.

\$STORAGE_CLASS

The Kubernetes storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: **oc get sc**. For example, for local storage:

```

databaseVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 120Gi
databaseLogsVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 12Gi
webVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 200Gi
backupVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 300Gi
adminVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 20Gi
certVolumeClaimTemplate:
  storageClassName: $STORAGE_CLASS
  volumeSize: 4Gi

```

Note:

- The `databaseVolumeClaimTemplate` has a minimum value of 30Gi. The `webVolumeClaimTemplate` has a minimum value of 20Gi. The `backupVolumeClaimTemplate` has a minimum value of 30Gi. However, all three need to be sized according to the installation.
- The `adminVolumeClaimTemplate` and `databaseLogsVolumeClaimTemplate` in this example have their minimum values and you do not need to change them.
- The `certVolumeClaimTemplate` is used by the nginx pod and in 2dcdr (two data center disaster recovery) mode it is also used by the tunnel, db-remote and www-remote pods. It can be sized small, just like the `adminVolumeClaimTemplate`, and does not need to be increased based on the number of sites.

\$LICENSE_ACCEPTANCE

Set `accept` to `true`. You must accept the license to successfully deploy API Connect.

\$LICENSE_USE

Set `use` to either `production` or `nonproduction` to match the license that you purchased.

\$LICENSE_ID

Set `license`: to the license ID for the version of API Connect that you purchased. See [API Connect licenses](#).

3. Optional: If you want to disable mTLS for communications between the management and portal, and enable JWT instead, then add and set the properties `mtlsValidateClient` and `jwtksUrl`.

```
spec:
  ...
  mtlsValidateClient: false
  jwtksUrl: <JWTKS URL>
```

where `<JWTKS URL>` is the URL of the JWKS endpoint hosted on the management subsystems. To find out the `jwtksUrl`, describe the management CR and check the `status`: section:

```
kubectl describe mgmt -n <namespace>
...
status:
- name: jwtksUrl
  secretName: api-endpoint
  type: API
  uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information on JWT security, see [Enable JWT security instead of mTLS](#).

4. Install the Portal subsystem by applying the modified CR with the following command:

```
oc apply -f portal_cr.yaml -n <namespace>
```

5. Verify that the Portal subsystem is fully installed:

```
oc get PortalCluster -n <namespace>
```

The installation is complete when `STATUS` reports `Running`, `Ready` reports that all services are online (3/3), and `MESSAGE` reports "Ready for API Cloud Manager service registration", for example:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	MESSAGE	AGE
portal	3/3	Running	<version>	<version>		Ready for API Cloud Manager service registration	2h

You do not need to wait for the portal installation to complete before you move on to the next subsystem installation.

6. Verify that you can access the portal web endpoint by browsing to the `portalUIEndpoint.hosts.name` as configured in your yaml file:

```
curl -k https://portal.$STACK_HOST
<!DOCTYPE html>
<html>
<head>
<title>Developer Portal is ready to create sites!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>Developer Portal is ready to create sites!</h1>
<p>If you see this page, the web server is successfully installed and
working. Further configuration is required.</p>
</body>
</html>
```

7. It is highly recommend that you retain the Portal subsystem encryption secret in a safe location, in case you need to restore the Portal subsystem during a disaster recovery scenario.

The encryption secret is created when the Portal subsystem is initially deployed. You can obtain the name of the encryption secret from the Portal subsystem custom resource after the installation is complete:

- a. Obtain the name of the encryptionSecret:

```
oc get ptl portal -o yaml -n <namespace> | grep encryptionSecret
encryptionSecret: portal-enc-key
```

The name of encryption secret is `portal-enc-key`.

- b. Save the secret yaml in a safe location:

```
oc get secret portal-enc-key -o yaml -n <namespace> > portal-enc-key.yaml
```

8. Backup the Portal installation. See [Backing up and restoring Developer Portal on OpenShift and Cloud Pak for Integration](#).

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the [Cloud Manager configuration checklist](#).

Previous topic: [Installing the Gateway subsystem](#)

Next topic: [Installing the Analytics subsystem](#)

Installing the Analytics subsystem

Install the Analytics subsystem by creating and applying the ingress-ca, common-issuer, and analytics_cr yaml files.

Before you begin

When installing subsystems in different namespaces you must first create the cert-manager issuers and certificates required by that subsystem in the same namespace.

1. Using the file obtained from [Extracting the Management ingress-ca certificates](#), run: `oc apply -f ingress-ca.yaml -n <analytics namespace>`
2. Create a file called `common-issuer.yaml` and paste in these contents:

Note: If you have just done [Installing the Portal subsystem](#) then you will already have this file.

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
```

```
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "selfsigning-issuer"
  }
spec:
  selfSigned: {}
---
```

```
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-issuer"
  }
spec:
  ca:
    secretName: ingress-ca
```

3. Apply the above yaml file with: `oc apply -f common-issuer.yaml -n <analytics namespace>`
4. Confirm that the issuers were created and are in ready state: `oc get issuers -n <analytics namespace>`

```
oc get issuers -n <analytics namespace>
NAME          READY
ingress-issuer  True
selfsigning-issuer  True
```

About this task

Use the OpenShift CLI to edit the custom resource template for the Analytics subsystem, apply the resource, verify that the pods are up and running.

Procedure

1. Create a file that is called `analytics_cr.yaml` and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
```

```

# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsCluster
metadata:
  name: analytics
  labels: {
    app.kubernetes.io/instance: "analytics",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "analytics"
  }
spec:
  version: $APP_PRODUCT_VERSION
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
  profile: $PROFILE
  microServiceSecurity: certManager
  certManagerIssuer:
    name: selfsigning-issuer
    kind: Issuer
  ingestion:
    endpoint:
      annotations:
        cert-manager.io/issuer: ingress-issuer
      hosts:
        - name: ai.$STACK_HOST
          secretName: analytics-ai-endpoint
    clientSubjectDN: CN=analytics-ingestion-client,O=cert-manager
  storage:
    type: shared # change to 'dedicated' if you want separate master and storage pods. Three-replica deployments only.
  shared:
    volumeClaimTemplate:
      storageClassName: $STORAGE_CLASS
      volumeSize: $DATA_VOLUME_SIZE
    # master: # uncomment this section if you set storage.type = dedicated.
    # volumeClaimTemplate:
    #   storageClassName: $STORAGE_CLASS

```

2. Edit the yaml file and set the variables:

```

$APP_PRODUCT_VERSION
API Connect application version for the subsystems.

version: <version_number>

Example version number: 10.0.5.4

```

\$PROFILE
Specify your analytics subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see [Planning your deployment topology](#).

\$STACK_HOST
The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can customize the subdomain or the complete hostname:

- Subdomain customization only
Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **\$STACK_HOST** with the desired ingress subdomain for the API Connect stack. For example, if your host is **myhost.subnet.example.com**:

```

ingestion:
  endpoint:
    < ... >
  hosts:
    - name: ai.myhost.subnet.example.com
      secret: analytics-ai-endpoint
    < ... >

```

- Complete hostname customization
Change both the predefined prefixes and the **\$STACK_HOST** subdomain to match your desired hostnames.

For example, you can replace **\$STACK_HOST** with **my.analytics.ingestion.myhost.subnet.example.com**, where **my.analytics.ingestion** replaces **ai**, and **myhost.subnet.example.com** replaces **STACK_HOST**.

```

ingestion:
  endpoint:
    < ... >
  hosts:
    - name: my.analytics.ingestion.myhost.subnet.example.com
      secret: analytics-ai-endpoint
    < ... >

```

\$STORAGE_CLASS

The Kubernetes storage class to be used for persistent volume claims (for more information, see [Planning the analytics profile, storage class, and storage type](#)). Find the available storage classes in the target cluster by running the following command: `oc get sc`. Example:

```
storage:
  type: shared
  shared:
    volumeClaimTemplate:
      storageClassName: ceph-block
```

`$DATA_VOLUME_SIZE`

Size of storage allocated for data. To estimate the storage space you require, see [Estimating internal storage space](#). If `$DATA_VOLUME_SIZE` is not specified, it defaults to 50Gi. See [Reference: Fields in the Analytics CR](#). Example:

```
storage:
  type: shared
  shared:
    volumeClaimTemplate:
      storageClassName: ceph-block
      volumeSize: 100Gi
```

`$LICENSE_ACCEPTANCE`

Set `accept` to `true`. You must accept the license to successfully deploy API Connect.

`$LICENSE_USE`

Set `use` to either `production` or `nonproduction` to match the license that you purchased.

`$LICENSE_ID`

Set `license` to the license ID for the version of API Connect that you purchased. See [API Connect licenses](#).

- Optional: Three replica deployments only: Enable dedicated storage.

If you want to use dedicated storage, update the `spec.storage` section of the yaml file as follows:

```
storage:
  type: dedicated
  shared:
    volumeClaimTemplate:
      storageClassName: $STORAGE_CLASS
      volumeSize: $DATA_VOLUME_SIZE
  master:
    volumeClaimTemplate:
      storageClassName: $STORAGE_CLASS
```

For more information about dedicated storage, see [Dedicated storage instead of shared storage](#).

- Optional: If you want to disable mTLS for communications between the management and analytics subsystem, and between the gateway and analytics subsystem, and enable JWT instead, then add and set the properties `mtlsValidateClient` and `jwtksUrl`.

```
spec:
  ...
  mtlsValidateClient: false
  jwtksUrl: <JWTKS URL>
```

where `<JWTKS URL>` is the URL of the JWKS endpoint hosted on the management subsystems. To find out the `jwtksUrl`, describe the management CR and check the `status` section:

```
kubectl describe mgmt -n <namespace>
...
status:
- name: jwtksUrl
  secretName: api-endpoint
  type: API
  uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information on JWT security, see [Enable JWT security instead of mTLS](#).

Note: It is not possible to use JWT on the V5 compatible gateway to analytics message flow.

- If you want to enable additional analytics features, you can set them in the `analytics_cr.yaml` file as described here: [Analytics subsystem CR configured settings](#)
- Install the analytics subsystem by applying the modified CR with the following command:

```
oc apply -f analytics_cr.yaml -n <namespace>
```

- Verify that the analytics subsystem is fully installed:

```
oc get AnalyticsCluster -n <namespace>
```

The installation is complete when `READY` shows all pods running (n/n), and the `STATUS` reports `Running`. Example:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
analytics	n/n	Running	10.0.5.3	10.0.5.3-1281	86m	

It is not necessary to wait for analytics installation to complete before you move on to the next subsystem installation.

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the [Cloud Manager configuration checklist](#).

Previous topic: [Installing the Portal subsystem](#)

Installing a two data center deployment with Cloud Pak for Integration

How to deploy a two data center disaster recovery (2DCDR) deployment on OpenShift with Cloud Pak for Integration.

Introduction

Ensure that you understand the concepts of 2DCDR in API Connect. For more information, see [Two data center deployment strategy on Kubernetes and OpenShift](#).

Familiarize yourself with the Cloud Pak for Integration installation instructions linked from [Deploying on OpenShift and Cloud Pak for Integration](#).

Note: During a 2DCDR installation, the management subsystem remains in a **Warning** state until the management subsystems in both data centers synchronize their databases. The management **Warning** status prevents the deployment and registration of the portal, gateway, and analytics subsystems. When the management subsystem databases are synchronized and both in **Ready** state, the deployment and registration of the other subsystems starts.

Prerequisites

The prerequisites for a 2DCDR deployment with Cloud Pak for Integration are as follows.

- The management and portal endpoints must be the same on both data centers.
- The network latency between data centers must be no more than 80 ms.
- The management cluster in each data center must use the same encryption secret.
- The portal cluster in each data center must use the same encryption secret.
- The ingress-ca certificate must be the same on both data centers.
- The Cloud Pak for Integration cluster must have a customized hostname and custom certificates.
- The API Connect namespace name must be the same in both clusters.
- The APICoordinateCluster CR name must be identical on both clusters.

Restrictions:

- API Connect is not supported on a FIPS-enabled environment.
- It is not possible to use the Automated API behavior testing application ([Installing the Automated API behavior testing application](#)) in a 2DCDR configuration ([Two data center deployment strategy on Kubernetes and OpenShift](#)).
- [Planning and initial preparation](#)
Decide on endpoint, host, and namespace names. Prepare your Cloud Pak for Integration environments for 2DCDR API Connect.
- [Preparing your active data center](#)
Create the secrets, certificates, and issuers that are needed for your active data center to replicate with the warm-standby.
- [Preparing your warm-standby data center](#)
Create the secrets, certificates, and issuers that are needed for your warm-standby data center to replicate with the active.
- [Installing API Connect on the active data center](#)
Follow the installation steps for API Connect described in the Cloud Pak for Integration documentation, adding the `multiSiteHA` configuration to the API Connect YAML.
- [Installing API Connect on the warm-standby data center](#)
Follow the installation steps for API Connect described in the Cloud Pak for Integration documentation, adding the `multiSiteHA` configuration to the API Connect YAML.

Planning and initial preparation

Decide on endpoint, host, and namespace names. Prepare your Cloud Pak for Integration environments for 2DCDR API Connect.

Procedure

1. Decide on naming. Both data centers must use the same names for:
 - API Connect namespace.
 - API Connect Cluster CR name. Also known as the API Connect instance name, and referred to in this document as `<apic-instance-name>`.
 - Portal subsystem endpoints
 - `portalAdminEndpoint`
 - `portalUIEndpoint`
 - Custom hostname and domain. This will be used in the URL for the Cloud Pak for Integration Platform UI and also for the Management subsystem endpoints. Traffic to this hostname should be directed to whichever data center is the active.

Important: Each data center must use a different backup path for your Management backups. For portal backups each data center must use the same backup path. For more information, see [Backup and restore requirements for a two data center deployment](#).
2. Create a namespace for your API Connect deployment in both data centers.
The API Connect namespace must be the same in both data centers, set it with:

```
oc create ns <namespace>
```
3. Install the Cloud Pak for Integration operator, Platform Navigator UI, and the API Connect operator in both your data centers. See <https://www.ibm.com/docs/en/cloud-paks/cp-integration>. Do not create an API Connect instance yet, install only the API Connect operator.
4. Decide which data center will start as the active and which will start as the warm-standby.
5. On the CLI of the active data center, where you run `oc` commands, create a directory called `2dcdcr-active-yamls`. On the CLI of the warm-standby data center, create a directory called `2dcdcr-ws-yamls`.

What to do next

Proceed to prepare the replication certificates, secrets, and issuers on your active data center [Preparing your active data center](#).

Preparing your active data center

Create the secrets, certificates, and issuers that are needed for your active data center to replicate with the warm-standby.

About this task

All operations are done on the CLI, in the `2dcdr-active-yamls` directory you created in [Planning and initial preparation](#).

In the yaml files and commands that are shown here, replace `<apic-instance-name>` with the name you intend to use for your API Connect Cluster CR, and `<namespace>` with your API Connect namespace name (which will be the same for both Management and Portal subsystems). As decided in [Planning and initial preparation](#).

Procedure

1. Deploy the cert-manager.

a. Create a file that is called `cert-manager.yaml` and paste in the following contents:

```
apiVersion: operator.ibm.com/v1alpha1
kind: OperandRequest
metadata:
  name: ibm-apiconnect
spec:
  requests:
    - operands:
        - name: ibm-cert-manager-operator
          registry: common-service
          registryNamespace: ibm-common-services
```

b. Apply this yaml file with:

```
oc apply -f cert-manager.yaml -n <namespace>
```

c. Confirm that the cert-manager is deployed:

```
oc get deploy -n ibm-common-services | grep ibm-cert-manager-operator
```

d. Wait for the cert-manager deployment to become available:

```
oc wait --for=condition=available deployment.apps/ibm-cert-manager-operator -n ibm-common-services --timeout=900s
```

e. Wait for the three cert-manager pods and operator to be in running state:

```
oc get pods -n ibm-common-services | grep cert
```

```
cert-manager-cainjector-566b5d5698-pzccn      1/1      Running    0      18m
cert-manager-controller-766ddf8c4-ftr85     1/1      Running    0      18m
cert-manager-webhook-79bb8f67b7-gvkst      1/1      Running    0      18m
ibm-cert-manager-operator-fdfd66bd4-6rrz6   1/1      Running    0      19m
```

2. Create the encryption key secrets for the Management and Portal subsystems.

a. Run the following command to create a file that contains a random string, which is used to create the management encryption key secret:

```
cat /dev/urandom | head -c63 | base64 -w0 > mgmt-enc-key.txt
```

b. Run the following command to create the management encryption key secret:

```
oc create secret generic mgmt-encryption-key --from-file=encryption_secret.bin=mgmt-enc-key.txt -n <management namespace>
```

c. Confirm that the secret was created successfully by running:

```
oc get secrets -n <management namespace> | grep mgmt-encryption-key
```

```
mgmt-encryption-key      Opaque      1      83s
```

d. Run the following command to create a file that contains a random string, which is used to create the portal encryption key secret:

```
cat /dev/urandom | head -c63 | base64 -w0 > ptl-enc-key.txt
```

e. Run the following command to create the portal encryption key secret:

```
oc create secret generic ptl-encryption-key --from-file=encryption_secret=ptl-enc-key.txt -n <portal namespace>
```

f. Confirm that the secret was created successfully by running:

```
oc get secrets -n <portal namespace> | grep ptl-encryption-key
```

```
ptl-encryption-key      Opaque      1      15s
```

3. Create a self-signed issuer.

a. Create a file that is called `issuer.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

```
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
```

```

  name: <apic-instance-name>-self-signed
spec:
  selfSigned: {}

```

b. Apply this yaml file with:

```
oc apply -f issuer.yaml -n <namespace>
```

c. Verify that the issuer was created with:

```
oc get issuer -n <namespace>
```

NAME	READY	AGE
apic-self-signed	True	25s

4. Create an ingress-ca certificate:

a. Create a file that is called `self-signed-issuer-cert.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

```

apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <apic-instance-name>-ingress-ca
spec:
  commonName: ingress-ca
  duration: 87600h0m0s
  isCA: true
  issuerRef:
    kind: Issuer
    name: <apic-instance-name>-self-signed
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: <apic-instance-name>-ingress-ca

```

b. Apply this yaml file with:

```
oc apply -f self-signed-issuer-cert.yaml -n <namespace>
```

c. Verify that the certificate was created with:

```
oc get cert -n <namespace>
```

NAME	READY	SECRET	AGE	EXPIRATION
apic-ingress-ca	True	apic-ingress-ca	11s	2032-08-15T13:01:47Z

5. Create the ingress issuer.

a. Create a file `ingress-issuer.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

```

apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: <apic-instance-name>-ingress-issuer
spec:
  ca:
    secretName: <apic-instance-name>-ingress-ca

```

b. Apply this yaml file with:

```
oc apply -f ingress-issuer.yaml -n <namespace>
```

c. Verify that the issuer was created with:

```
oc get issuer -n <namespace>
```

NAME	READY	AGE
apic-ingress-issuer	True	20s

6. Create a custom hostname and certificate for Cloud Pak for Integration Platform UI and IBM Cloud Pak foundational services.

a. Create a file `custom-hostname-cert.yaml` and paste in the following contents:

```

apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <apic-instance-name>-custom-hostname-cert
  namespace: <Platform UI namespace>
spec:
  secretName: <apic-instance-name>-custom-hostname-secret
  issuerRef:
    name: <apic-instance-name>-ingress-issuer
    kind: Issuer
  commonName: <custom fqdn>
  duration: 17520h0m0s
  renewBefore: 720h0m0s
  dnsNames:
  - <ocp cluster api address>
  - <ocp cluster ui address>
  - <cp console ui address>

```

Where:

- `<apic-instance-name>` is the API Connect Cluster CR name.
- `<platform ui namespace>` is the namespace used by your Cloud Pak for Integration Platform UI.
- `<custom fqdn>` is the fully qualified custom hostname.
- `<ocp cluster api address>` is the fully qualified API address of your Cloud Pak for Integration cluster.
- `<ocp cluster ui address>` is the fully qualified UI address of your Cloud Pak for Integration cluster.

- `<cp console ui address>` is the fully qualified address of your Cloud Pak console. Run the following command to see this address: `oc get route -n ibm-common-services cp-console -o jsonpath='{.spec.host}'`.

b. Apply this file with:

```
oc apply -f custom-hostname-cert.yaml -n <namespace>
```

c. Extract the generated certificate files with:

```
oc extract secret/<apic-instance-name>-custom-hostname-secret -n <namespace> --to=. --keys=tls.crt,tls.key,ca.crt --confirm
```

Check that your local directory contains these three extracted files:

```
ca.crt
tls.crt
tls.key
```

d. Follow the steps in the Cloud Pak for Integration documentation to create a custom hostname and certificate <https://www.ibm.com/docs/en/cloud-paks/cp-integration>. For Cloud Pak for Integration v2022.2 the steps are on this page: [Creating a custom hostname and certificate](#). Use the certificate files you extracted in the previous step.

e. Create a secret in the Platform UI namespace using the same certificates you created in [6.c](#).

```
oc -n ${PLATFORM_UI_NAMESPACE} create secret generic route-tls-secret --from-file=ca.crt=ca.crt --from-file=tls.crt=tls.crt --from-file=tls.key=tls.key
```

f. Add this new secret to the Platform UI CR, as described in the Cloud Pak for Integration documentation on using custom hostnames and certificates for the Platform UI. For Cloud Pak for Integration v2022.2 the steps are on this page: <https://www.ibm.com/docs/en/cloud-paks/cp-integration/2022.2?topic=certificates-using-custom-hostnames-platform-ui>.

7. Create the TLS client replication certificates for Management and Portal.

a. Create a yaml file that is called `mgmt-tls-client-cert.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <apic-instance-name>-mgmt-replication-client
spec:
  commonName: <apic-instance-name>-mgmt-replication-client
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: <apic-instance-name>-ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: <apic-instance-name>-mgmt-replication-client
```

b. Apply this yaml file with:

```
oc apply -f mgmt-tls-client-cert.yaml -n <management namespace>
```

c. Verify that the certificate was created with:

```
oc get certs -n <management namespace>
```

NAME	READY	SECRET	AGE
EXPIRATION			
...			
<apic-instance-name>-mgmt-replication-client	True	<apic-instance-name>-mgmt-replication-client	16m 2024-08-17T13:04:27Z

d. Create a file `ptl-tls-client-cert.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <apic-instance-name>-ptl-replication-client
spec:
  commonName: <apic-instance-name>-ptl-replication-client
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: <apic-instance-name>-ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: <apic-instance-name>-ptl-replication-client
```

e. Apply this yaml file with:

```
oc apply -f ptl-tls-client-cert.yaml -n <portal namespace>
```

f. Verify that the certificate was created with:

```
oc get certs -n <portal namespace>
```

NAME	READY	SECRET	AGE
EXPIRATION			
...			
<apic-instance-name>-ptl-replication-client	True	<apic-instance-name>-ptl-replication-client	16m 2024-08-17T13:04:27Z

8. Export the ingress-ca issuer secret.

The ingress-ca issuer secret must be the same on the warm-standby data center. Export it from the active so that it can be imported on the warm-standby data center.

- a. Run the following command to export the secret to a file called `ca-issuer-secret.yaml`:

```
oc get secret <apic-instance-name>-ingress-ca -o yaml -n <namespace> > ca-issuer-secret.yaml
```

- b. Edit the `ca-issuer-secret.yaml` file to remove the `creationTimestamp`, `resourceVersion`, `uid`, and `managedFields`. Remove the labels and annotations sections completely. The contents should look like this:

```
apiVersion: v1
data:
  ca.crt: <long cert string>
  tls.crt: <long cert string>
  tls.key: <long cert string>
kind: Secret
metadata:
  name: <apic-instance-name>-ingress-ca
  namespace: <namespace>
type: kubernetes.io/tls
```

9. Copy the following files from your `2dcd-active-yamls` directory to the `2dcd-ws-yamls` directory in your warm-standby data center.

```
ca-issuer-secret.yaml
mgmt-enc-key.txt
ptl-enc-key.txt
```

These files are required to ensure that the ingress-ca and encryption secrets on both sites are the same.

Tip: To save time during the preparation of your warm-standby data center you can also copy these yaml files:

```
ptl-tls-client-cert.yaml
mgmt-tls-client-cert.yaml
ingress-issuer.yaml
```

What to do next

Create your secrets, certs, and issuers on your warm-standby data center: [Preparing your warm-standby data center](#).

Preparing your warm-standby data center

Create the secrets, certificates, and issuers that are needed for your warm-standby data center to replicate with the active.

About this task

All operations are done on the CLI, in the `2dcd-ws-yamls` directory you created in [Planning and initial preparation](#).

In the yaml files and commands that are shown here, replace `<apic-instance-name>` with the name you intend to use for your API Connect Cluster CR, and `<namespace>` with your API Connect namespace name (which will be the same for both Management and Portal subsystems). As decided in [Planning and initial preparation](#).

Procedure

1. Deploy the cert-manager.
 - a. Create a file that is called `cert-manager.yaml` and paste in the following contents:

```
apiVersion: operator.ibm.com/v1alpha1
kind: OperandRequest
metadata:
  name: ibm-apiconnect
spec:
  requests:
  - operands:
    - name: ibm-cert-manager-operator
      registry: common-service
      registryNamespace: ibm-common-services
```

- b. Apply this yaml file with:

```
oc apply -f cert-manager.yaml -n <namespace>
```

- c. Confirm that the cert-manager is deployed:

```
oc get deploy -n ibm-common-services | grep ibm-cert-manager-operator
```

- d. Wait for the cert-manager deployment to become available:

```
oc wait --for=condition=available deployment.apps/ibm-cert-manager-operator -n ibm-common-services --timeout=900s
```

- e. Wait for the three cert-manager pods and operator to be in running state:

```
oc get pods -n ibm-common-services | grep cert
```

```
cert-manager-cainjector-566b5d5698-pzccn      1/1      Running    0      18m
cert-manager-controller-766dff8c4-ftr85     1/1      Running    0      18m
cert-manager-webhook-79bb8f67b7-gvkst       1/1      Running    0      18m
ibm-cert-manager-operator-fdfd66bd4-6rrz6   1/1      Running    0      19m
```

2. Import the ingress-ca issuer secret.

This secret was created on your active data center here: [Export the ingress-ca issuer secret](#)

- Copy the `ca-issuer-secret.yaml` file to your warm-standby data center (if not done already).
- Create the secret with:

```
oc apply -f ca-issuer-secret.yaml -n <namespace>
```

- Verify that the secret was created with:

```
oc get secret -n <namespace>
```

NAME	TYPE	DATA	AGE
<apic-instance-name>-ingress-ca	kubernetes.io/tls	3	19h

3. Create the ingress issuer.

- Create a file `ingress-issuer.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

```
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: <apic-instance-name>-ingress-issuer
spec:
  ca:
    secretName: <apic-instance-name>-ingress-ca
```

- Apply this yaml file with:

```
oc apply -f ingress-issuer.yaml -n <namespace>
```

- Verify that the issuer was created with:

```
oc get issuer -n <namespace>
```

NAME	READY	AGE
apic-ingress-issuer	True	20s

4. Create a custom hostname and certificate for Cloud Pak for Integration Platform UI and IBM Cloud Pak foundational services.

- Create a file `custom-hostname-cert.yaml` and paste in the following contents:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <apic-instance-name>-custom-hostname-cert
  namespace: <Platform UI namespace>
spec:
  secretName: <apic-instance-name>-custom-hostname-secret
  issuerRef:
    name: <apic-instance-name>-ingress-issuer
    kind: Issuer
  commonName: <custom fqdn>
  duration: 17520h0m0s
  renewBefore: 720h0m0s
  dnsNames:
  - <ocp cluster api address>
  - <ocp cluster ui address>
  - <cp console ui address>
```

Where:

- `<apic-instance-name>` is the API Connect Cluster CR name.
- `<platform ui namespace>` is the namespace used by your Cloud Pak for Integration Platform UI.
- `<custom fqdn>` is the fully qualified custom hostname.
- `<ocp cluster api address>` is the fully qualified API address of your Cloud Pak for Integration cluster.
- `<ocp cluster ui address>` is the fully qualified UI address of your Cloud Pak for Integration cluster.
- `<cp console ui address>` is the fully qualified address of your Cloud Pak console. Run the following command to see this address: `oc get route -n ibm-common-services cp-console -o jsonpath='{.spec.host}'`.

- Apply this file with:

```
oc apply -f custom-hostname-cert.yaml -n <namespace>
```

- Extract the generated certificate files with:

```
oc extract secret/<apic-instance-name>-custom-hostname-secret -n <namespace> --to=. --keys=tls.crt,tls.key,ca.crt --confirm
```

Check that your local directory contains these three extracted files:

```
ca.crt
tls.crt
tls.key
```

- Follow the steps in the Cloud Pak for Integration documentation to create a custom hostname and certificate <https://www.ibm.com/docs/en/cloud-paks/cp-integration>. For Cloud Pak for Integration v2022.2 the steps are on this page: [Creating a custom hostname and certificate](#). Use the certificate files you extracted in the previous step.
- Create a secret in the Platform UI namespace using the same certificates you created in [4.c](#).

```
oc -n ${PLATFORM_UI_NAMESPACE} create secret generic route-tls-secret --from-file=ca.crt=ca.crt --from-file=tls.crt=tls.crt --from-file=tls.key=tls.key
```

- Add this new secret to the Platform UI CR, as described in the Cloud Pak for Integration documentation on using custom hostnames and certificates for the Platform UI. For Cloud Pak for Integration v2022.2 the steps are on this page: <https://www.ibm.com/docs/en/cloud-paks/cp-integration/2022.2?topic=certificates-using-custom-hostnames-platform-ui>.

5. Create the encryption key secrets for the Management and Portal subsystems, by using the random string generated on the active data center. Use the `mgmt-enc-key.txt` and `ptl-enc-key.txt` files that were copied to your warm-standby data center here: [Copy keys to warm-standby](#).
 - a. Run the following command to create the management encryption key secret:

```
oc create secret generic mgmt-encryption-key --from-file=encryption_secret.bin=mgmt-enc-key.txt -n <management namespace>
```

- b. Confirm that the secret was created successfully by running:

```
oc get secrets -n <management namespace> | grep mgmt-encryption-key

mgmt-encryption-key      Opaque      1      83s
```

- c. Run the following command to create the portal encryption key secret:

```
oc create secret generic ptl-encryption-key --from-file=encryption_secret=ptl-enc-key.txt -n <portal namespace>
```

- d. Confirm that the secret was created successfully by running:

```
oc get secrets -n <portal namespace> | grep ptl-encryption-key

ptl-encryption-key      Opaque      1      15s
```

6. Create the TLS client replication certificates for Management and Portal.

- a. Create a yaml file that is called `mgmt-tls-client-cert.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <apic-instance-name>-mgmt-replication-client
spec:
  commonName: <apic-instance-name>-mgmt-replication-client
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: <apic-instance-name>-ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: <apic-instance-name>-mgmt-replication-client
```

- b. Apply this yaml file with:

```
oc apply -f mgmt-tls-client-cert.yaml -n <management namespace>
```

- c. Verify that the certificate was created with:

```
oc get certs -n <management namespace>
```

NAME	READY	SECRET	AGE
EXPIRATION			
...			
<apic-instance-name>-mgmt-replication-client	True	<apic-instance-name>-mgmt-replication-client	16m 2024-08-17T13:04:27Z

- d. Create a file `ptl-tls-client-cert.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <apic-instance-name>-ptl-replication-client
spec:
  commonName: <apic-instance-name>-ptl-replication-client
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: <apic-instance-name>-ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: <apic-instance-name>-ptl-replication-client
```

- e. Apply this yaml file with:

```
oc apply -f ptl-tls-client-cert.yaml -n <portal namespace>
```

- f. Verify that the certificate was created with:

```
oc get certs -n <portal namespace>
```

NAME	READY	SECRET	AGE
EXPIRATION			
...			
<apic-instance-name>-ptl-replication-client	True	<apic-instance-name>-ptl-replication-client	16m 2024-08-17T13:04:27Z

What to do next

Install API Connect in your active data center: [Installing API Connect on the active data center](#).

Installing API Connect on the active data center

Follow the installation steps for API Connect described in the Cloud Pak for Integration documentation, adding the `multiSiteHA` configuration to the API Connect YAML.

Before you begin

Verify that all the secrets, certificates, and issuers are ready:

```
oc get secrets -n <namespace> | grep <apic-instance-name>
<apic-instance-name>-ingress-ca          kubernetes.io/tls      3      3d21h
<apic-instance-name>-mgmt-replication-client  kubernetes.io/tls      3      3d21h
<apic-instance-name>-ptl-replication-client  kubernetes.io/tls      3      3d21h

oc get certs -n <namespace> | grep <apic-instance-name>
<apic-instance-name>-ingress-ca          True <apic-instance-name>-ingress-ca      3d21h  2032-08-15T13:01:47Z
<apic-instance-name>-mgmt-replication-client  True <apic-instance-name>-mgmt-replication-client  3d21h  2024-08-17T13:04:27Z
<apic-instance-name>-ptl-replication-client  True <apic-instance-name>-ptl-replication-client  3d21h  2024-08-17T13:04:26Z

oc get issuer -n <namespace> | grep <apic-instance-name>
<apic-instance-name>-ingress-issuer      True  3d21h
<apic-instance-name>-self-signed         True  3d21h
```

Where `<apic-instance-name>` is the name you intend to use for your API Connect cluster CR, and `<namespace>` is the namespace you created for API Connect.

Ensure that your network is redirecting the custom hostname to the Platform UI in your active data center.

About this task

In the yaml files that are shown here, replace `<apic-instance-name>` with the name you intend to use for your API Connect Cluster CR. As decided in [Planning and initial preparation](#). Set `<active data center ingress domain>` and `<warm-standby data center ingress domain>` to their appropriate values, which can be determined by running this command in each data center:

```
oc get ingresses.config/cluster -o jsonpath={.spec.domain}
```

Procedure

1. Follow the API Connect installation steps using the Platform UI, as described in the Cloud Pak for Integration documentation <https://www.ibm.com/docs/en/cloud-paks/cp-integration>. At the point where you specify the configuration properties of your API Connect instance, switch to the YAML tab so you can edit the YAML directly.

Remember: Use the same `<apic-instance-name>` value when you specify the API Connect instance name in the Cloud Pak for Integration Platform UI.

2. Add the management `multiSiteHA` section to the YAML file under the `spec:` section.

```
management:
  encryptionSecret:
    secretName: mgmt-encryption-key
  multiSiteHA:
    mode: active
  replicationEndpoint:
    annotations:
      cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
    hosts:
      - name: mgmt-replication.<active data center ingress domain>
        secretName: <apic-instance-name>-mgmt-replication-server
    replicationPeerFQDN: mgmt-replication.<warm-standby data center ingress domain>
  tlsClient:
    secretName: <apic-instance-name>-mgmt-replication-client
```

3. Add the portal `multiSiteHA` section to the YAML file under the `spec:` section:

```
portal:
  portalAdminEndpoint:
    annotations:
      cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
    hosts:
      - name: <external load balanced portal admin hostname>
        secretName: portal-admin
  portalUIEndpoint:
    annotations:
      cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
    hosts:
      - name: <external load balanced portal web hostname>
        secretName: portal-web
  encryptionSecret:
    secretName: ptl-encryption-key
  multiSiteHA:
    mode: active
  replicationEndpoint:
    annotations:
      cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
    hosts:
      - name: ptl-replication.<active data center ingress domain>
        secretName: <apic-instance-name>-ptl-replication-server
    replicationPeerFQDN: ptl-replication.<warm-standby data center ingress domain>
  tlsClient:
    secretName: <apic-instance-name>-ptl-replication-client
```

4. Continue the installation steps as described in [Deploying on OpenShift and Cloud Pak for Integration](#).

Results

Confirm that the management subsystem is ready, but in **Warning** state with `oc get`

`mgmt`:

```
oc get mgmt -n <namespace>
```

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	MESSAGE
management	n/n	Warning	10.0.5.3-0	10.0.5.3-0		Management is ready. HA Status Warning - see HAStatus in CR for details

```
status:
```

```
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication not working, install or upgrade in progress.",
    "reason": "na",
    "status": "True",
    "type": "Pending"
  }
```

The management CR is expected to report the status of **Warning** until the warm-standby management subsystem is deployed, and both management subsystems complete data replication. When you see the status message "Management is ready. HA Status Warning - see HAStatus in CR for details", you can move on to [Installing API Connect on the warm-standby data center](#).

Note: The portal, analytics, and gateway subsystems are not deployed until the management replication is complete and the management CRs in both data centers report the following:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	MESSAGE
management	n/n	Running	10.0.5.3-0	10.0.5.3-0		Management is ready. 8m59s

```
status:
```

```
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication is working",
    "reason": "na",
    "status": "True",
    "type": "Ready"
  }
```

What to do next

Install API Connect on your warm-standby data center: [Installing API Connect on the warm-standby data center](#).

Installing API Connect on the warm-standby data center

Follow the installation steps for API Connect described in the Cloud Pak for Integration documentation, adding the `multiSiteHA` configuration to the API Connect YAML.

Before you begin

Verify that all the secrets, certificates, and issuers are ready:

```
oc get certificate -n <namespace> | grep <apic-instance-name>
NAME                                READY  SECRET  AGE  EXPIRATION
apic-mgmt-replication-client        True   apic-mgmt-replication-client  57s  2024-08-24T09:45:03Z
apic-ptl-replication-client         True   apic-ptl-replication-client   45s  2024-08-24T09:45:17Z

[root@api.iaincp4ilarge2.cp.fyre.ibm.com ~]# oc get secrets -n <namespace> | grep <apic-instance-name>
NAME                                TYPE  DATA  AGE
apic-ingress-ca                     kubernetes.io/tls  3  4m6s
apic-mgmt-replication-client        kubernetes.io/tls  3  2m40s
apic-ptl-replication-client         kubernetes.io/tls  3  2m26s

[root@api.iaincp4ilarge2.cp.fyre.ibm.com ~]# oc get issuer -n <namespace> | grep <apic-instance-name>
NAME                                READY  AGE
apic-ingress-issuer                 True   4m8s
```

Where `<apic-instance-name>` is the name you intend to use for your API Connect cluster CR, and `<namespace>` is the namespace you created for API Connect.

Ensure that your network is redirecting the custom hostname to the Platform UI in your warm-standby data center.

About this task

In the yml files that are shown here, replace `<apic-instance-name>` with the name you intend to use for your API Connect Cluster CR. As decided in [Planning and initial preparation](#). Set `<active data center ingress domain>` and `<warm-standby data center ingress domain>` to their appropriate values, which can be determined by running this command in each data center:

```
oc get ingresses.config/cluster -o jsonpath={.spec.domain}
```

Procedure

- Follow the API Connect installation steps using the Platform UI, as described in the Cloud Pak for Integration documentation <https://www.ibm.com/docs/en/cloud-paks/cp-integration>. At the point where you specify the configuration properties of your API Connect instance, switch to the YAML tab so you can edit the YAML directly.
Remember: Use the same `<apic-instance-name>` value when you specify the API Connect instance name in the Cloud Pak for Integration Platform UI.
- Add the management `multiSiteHA` section to the YAML file under the `spec` section.

```
management:
  encryptionSecret:
    secretName: mgmt-encryption-key
  multiSiteHA:
    mode: passive
    replicationEndpoint:
      annotations:
        cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
      hosts:
        - name: mgmt-replication.<warm-standby data center ingress domain>
          secretName: <apic-instance-name>-mgmt-replication-server
    replicationPeerFQDN: mgmt-replication.<active data center ingress domain>
    tlsClient:
      secretName: <apic-instance-name>-mgmt-replication-client
```

Note: Warm-standby is referred to as 'passive' in the CR YAML.

- Add the portal `multiSiteHA` section to the YAML file under the `spec` section:

```
portal:
  portalAdminEndpoint:
    annotations:
      cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
    hosts:
      - name: <external load balanced portal admin hostname>
        secretName: portal-admin
  portalUIEndpoint:
    annotations:
      cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
    hosts:
      - name: <external portal load balanced web hostname>
        secretName: portal-web
  encryptionSecret:
    secretName: ptl-encryption-key
  multiSiteHA:
    mode: passive
    replicationEndpoint:
      annotations:
        cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
      hosts:
        - name: ptl-replication.<warm-standby data center ingress domain>
          secretName: <apic-instance-name>-ptl-replication-server
    replicationPeerFQDN: ptl-replication.<active data center ingress domain>
    tlsClient:
      secretName: <apic-instance-name>-ptl-replication-client
```

- Disable the configurator service, as this will have already run on the active site.
 - Add the following section to the API Connect cluster CR:

```
disabledServices:
  - configurator
```

- Continue the installation steps as described in [Deploying on OpenShift and Cloud Pak for Integration](#).

Results

Installation can take around 1 hour to complete.

While the management subsystems on the warm-standby and active data centers are synchronizing, the management status reports **Warning**, and the `haStatus` reports **pending**:

```
oc get mgmt -n <namespace>
```

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	MESSAGE
management	n/n	Warning	10.0.5.3-0	10.0.5.3-0		Management is ready. HA Status Warning - see HAStatus in CR for details

```
status:
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication not working, install or upgrade in progress.",
    "reason": "na",
    "status": "True",
    "type": "Pending"
  }
```

When the management database replication between sites is complete, the management status reports **Running**, and `status.haStatus` reports **Ready**:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	MESSAGE
management	n/n	Running	10.0.5.3-0	10.0.5.3-0		Management is ready. 8m59s

```
status:
  haStatus
  {
```

```

"lastTransitionTime": "2023-03-31T19:47:08Z",
"message": "Replication is working",
"reason": "na",
"status": "True",
"type": "Ready"
}

```

After the management subsystems in the active and warm-standby sites are synchronized, the other subsystems are deployed. Verify on both active and warm-standby sites that all subsystems reach **Running** state:

```
oc get all -n <namespace>
```

```

...

```

NAME	READY	STATUS	VERSION	RECONCILED VERSION	AGE
analyticscluster.analytics.apiconnect.ibm.com/apis-minimum-a7s	n/n	Running	10.0.5.3	10.0.5.3-1281	27m
NAME	READY	STATUS	VERSION	RECONCILED VERSION	AGE
apiconnectcluster.apiconnect.ibm.com/apis-minimum	n/n	Ready	10.0.5.3	10.0.5.3-1281	2d20h
NAME	PHASE	READY	SUMMARY	VERSION	AGE
datapowerservice.datapower.ibm.com/apis-minimum-gw	Running	True	StatefulSet replicas ready: 1/1	10.5.0.0	25m
NAME	PHASE	LAST EVENT	WORK PENDING	WORK IN-PROGRESS	AGE
datapowermonitor.datapower.ibm.com/apis-minimum-gw	Running		false	false	25m
NAME	READY	STATUS	VERSION	RECONCILED VERSION	AGE
gatewaycluster.gateway.apiconnect.ibm.com/apis-minimum-gw	n/n	Running	10.0.5.3	10.0.5.3-1281	27m
NAME	READY	STATUS	VERSION	RECONCILED VERSION	AGE
managementcluster.management.apiconnect.ibm.com/apis-minimum-mgmt	8/8	Running	10.0.5.3	10.0.5.3-1281	2d20h
NAME	READY	STATUS	VERSION	RECONCILED VERSION	AGE
portalcluster.portal.apiconnect.ibm.com/apis-minimum-ptl	n/n	Running	10.0.5.3	10.0.5.3-1281	27m

Note: The warm-standby site has fewer **managementcluster** pods than the active site.

If the management database replication between sites fails for any reason other than because an install or upgrade is in progress, the **haStatus** output shows:

NAME	READY	STATUS	VERSION	RECONCILED VERSION	MESSAGE
management	n/n	Warning	10.0.5.3-0	10.0.5.3-0	Management is ready. HA Status Warning - see HAStatus in CR for details

```

status:
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication not working",
    "reason": "na",
    "status": "True",
    "type": "Warning"
  }

```

If the warning persists, see [Troubleshooting a two data center deployment](#).

You can validate that your portal deployments are synchronizing by running `oc get pods` on both the active and warm-standby data centers. Confirm that the number and names of the pods all match (the UUIDs in the names might be different on each site), and that all are in the **Ready** state.

For additional replication verification checks, see [Verifying replication between data centers](#). It is recommended to run a test failover, and confirm that all of the expected data is present and correct on the newly active site. See [How to failover API Connect from the active to the warm-standby data center](#) for details.

What to do next

Configure your deployment: [Cloud Manager configuration checklist](#).

Important: It is strongly recommended to complete the disaster recovery preparation on both sites: [Disaster recovery on OpenShift and Cloud Pak for Integration](#). Disaster recovery preparation ensures that you can recover your API Connect installation if both data centers are lost, or if a replication failure occurs. Both sites must use different backup paths for their Management and Portal backups.

Installing a two data center deployment on OpenShift

How to deploy a two data center disaster recovery (2DCDR) deployment on OpenShift.

Introduction

Ensure that you understand the concepts of 2DCDR in API Connect. For more information, see [Two data center deployment strategy on Kubernetes and OpenShift](#).

Familiarize yourself with the OpenShift installation instructions linked from [Deploying on OpenShift and Cloud Pak for Integration](#).

Note: 2DCDR is not supported on OpenShift where the top-level CR is used. For 2DCDR on OpenShift, you must install API Connect as individual subsystems. See [Installing with subsystem CRs in a shared namespace on OpenShift](#) and [Installing with subsystem CRs in different namespaces or environments on OpenShift](#).

Prerequisites

The prerequisites for a 2DCDR deployment on OpenShift are as follows.

- The management and portal endpoints must be the same on both data centers.
- The network latency between data centers must be no more than 80 ms.

- The management cluster in each data center must use the same encryption secret.
- The portal cluster in each data center must use the same encryption secret.
- The ingress-ca certificate must be the same on both data centers.

Restrictions:

- API Connect is not supported on a FIPS-enabled environment.
- It is not possible to use the Automated API behavior testing application ([Installing the Automated API behavior testing application](#)) in a 2DCDR configuration ([Two data center deployment strategy on Kubernetes and OpenShift](#)).
- [Planning and initial preparation](#)
Decide on endpoint names. Prepare your OpenShift environments for 2DCDR API Connect.
- [Preparing your active data center](#)
Create the secrets, certificates, and issuers that are needed for your active data center to replicate with the warm-standby.
- [Preparing your warm-standby data center](#)
Create the secrets, certificates, and issuers that are needed for your warm-standby data center to replicate with the active.
- [Installing API Connect on the active data center](#)
Add the `multiSiteHA` configuration to the API Connect Management and Portal CR YAML files before they are applied.
- [Installing API Connect on the warm-standby data center](#)
Add the `multiSiteHA` configuration to the API Connect Management and Portal YAML files before they are applied.

Planning and initial preparation

Decide on endpoint names. Prepare your OpenShift environments for 2DCDR API Connect.

Procedure

1. Decide on naming. Both data centers must use the same names for:
 - Management subsystem endpoints
 - `cloudManagerEndpoint`
 - `apiManagerEndpoint`
 - `platformAPIEndpoint`
 - `consumerAPIEndpoint`
 - Portal subsystem endpoints
 - `portalAdminEndpoint`
 - `portalUIEndpoint`

Each data center must have a different `siteName`. For example, `dallas` in data center 1, and `raleigh` in data center 2. The site name can contain only `a-z` and `0-9` characters. You set `siteName` in the `ManagementCluster` and `PortalCluster` CRs when the subsystems are first deployed. You cannot change the name after deployment. If you don't specify `siteName` at first deployment, a random `siteName` is created for you.

Important: Each data center must use a different backup path for your Management backups. For portal backups each data center must use the same backup path. For more information, see [Backup and restore requirements for a two data center deployment](#).

2. Install the API Connect operators in both your data centers. See [Installing operators for a single namespace API Connect installation](#) or [Installing operators for a multi namespace API Connect installation](#), depending on whether all your API Connect subsystems are to be in the same namespace or not. Return here when the operators are installed.
3. If not already done, create namespaces for your API Connect subsystems in both data centers.

```
oc create ns <namespace>
```

the data centers do not have to use the same namespace name.

4. Decide which data center will start as the active and which will start as the warm-standby.
5. On the CLI of the active data center, where you run `oc` commands, create a directory called `2dcd-active-yamls`. On the CLI of the warm-standby data center, create a directory called `2dcd-ws-yamls`.

What to do next

Proceed to prepare the replication certificates, secrets, and issuers on your active data center [Preparing your active data center](#).

Preparing your active data center

Create the secrets, certificates, and issuers that are needed for your active data center to replicate with the warm-standby.

About this task

All operations are done on the CLI, in the `2dcd-active-yamls` directory you created in [Planning and initial preparation](#).

In the yaml files and commands that are shown here, replace `<namespace>` with the name of the corresponding subsystem namespace.

Procedure

1. Deploy the cert-manager.
 - a. Create a file that is called `cert-manager.yaml` and paste in the following contents:

```

apiVersion: operator.ibm.com/v1alpha1
kind: OperandRequest
metadata:
  name: ibm-apiconnect
spec:
  requests:
  - operands:
    - name: ibm-cert-manager-operator
      registry: common-service
      registryNamespace: ibm-common-services

```

b. Apply this yaml file with:

```
oc apply -f cert-manager.yaml -n <namespace>
```

c. Confirm that the cert-manager is deployed:

```
oc get deploy -n ibm-common-services | grep ibm-cert-manager-operator
```

d. Wait for the cert-manager deployment to become available:

```
oc wait --for=condition=available deployment.apps/ibm-cert-manager-operator -n ibm-common-services --timeout=900s
```

e. Wait for the three cert-manager pods and operator to be in running state:

```
oc get pods -n ibm-common-services | grep cert
```

```

cert-manager-cainjector-566b5d5698-pzccn      1/1      Running    0          18m
cert-manager-controller-766ddf8c4-ftr85     1/1      Running    0          18m
cert-manager-webhook-79bb8f67b7-gvkst      1/1      Running    0          18m
ibm-cert-manager-operator-fdfd66bd4-6rrz6   1/1      Running    0          19m

```

2. Create the encryption key secrets for the Management and Portal subsystems.

a. Run the following command to create a file that contains a random string, which is used to create the management encryption key secret:

```
cat /dev/urandom | head -c63 | base64 -w0 > mgmt-enc-key.txt
```

b. Run the following command to create the management encryption key secret:

```
oc create secret generic mgmt-encryption-key --from-file=encryption_secret.bin=mgmt-enc-key.txt -n <management namespace>
```

c. Confirm that the secret was created successfully by running:

```
oc get secrets -n <management namespace> | grep mgmt-encryption-key
```

```
mgmt-encryption-key      Opaque      1      83s
```

d. Run the following command to create a file that contains a random string, which is used to create the portal encryption key secret:

```
cat /dev/urandom | head -c63 | base64 -w0 > ptl-enc-key.txt
```

e. Run the following command to create the portal encryption key secret:

```
oc create secret generic ptl-encryption-key --from-file=encryption_secret=ptl-enc-key.txt -n <portal namespace>
```

f. Confirm that the secret was created successfully by running:

```
oc get secrets -n <portal namespace> | grep ptl-encryption-key
```

```
ptl-encryption-key      Opaque      1      15s
```

3. If all your API Connect subsystems are to be installed in the same namespace, then create your certificates, issuers, and secrets as described here: [Setting up a certificate issuer](#). Return here when complete, do not proceed to install the subsystems.

4. If your API Connect subsystems are to be installed in different namespaces, then create your certificates, issuers, and secrets for your Management subsystem as described here [Create Management subsystem issuers and secrets](#). For your Portal subsystem, follow the steps here [Create Portal subsystem issuers and secrets](#). Return here when complete, do not proceed to install either subsystem.

5. Create the TLS client replication certificates for Management and Portal.

a. Create a yaml file that is called `mgmt-tls-client-cert.yaml` and paste in the following contents:

```

apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: mgmt-replication-client
spec:
  commonName: mgmt-replication-client
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: mgmt-replication-client

```

b. Apply this yaml file with:

```
oc apply -f mgmt-tls-client-cert.yaml -n <management namespace>
```

c. Verify that the certificate was created with:

```
oc get certs -n <management namespace>
```

```

NAME                                READY    SECRET    AGE    EXPIRATION

```

```
...
mgmt-replication-client True mgmt-replication-client 16m 2024-08-17T13:04:27Z
```

d. Create a file `ptl-tls-client-cert.yaml` and paste in the following contents:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ptl-replication-client
spec:
  commonName: ptl-replication-client
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: ptl-replication-client
```

e. Apply this yaml file with:

```
oc apply -f ptl-tls-client-cert.yaml -n <portal namespace>
```

f. Verify that the certificate was created with:

```
oc get certs -n <portal namespace>
```

```
NAME                                READY  SECRET                                AGE  EXPIRATION
...
ptl-replication-client              True   ptl-replication-client              16m  2024-08-17T13:04:27Z
```

6. Export the ingress-ca issuer secret.

The ingress-ca issuer secret must be the same on the warm-standby data center. Export it from the active so that it can be imported on the warm-standby data center.

Note: If you are deploying API Connect subsystems in different namespaces, then you export the Management subsystem ingress-ca secret.

a. Run the following command to export the secret to a file called `ca-issuer-secret.yaml`:

```
oc get secret ingress-ca -o yaml -n <management namespace> > ca-issuer-secret.yaml
```

b. Edit the `ca-issuer-secret.yaml` file to remove the creationTimestamp, resourceVersion, uid, namespace, and managedFields. Remove the labels and annotations sections completely. The resulting contents should look like this:

```
apiVersion: v1
data:
  ca.crt: <long cert string>
  tls.crt: <long cert string>
  tls.key: <long cert string>
kind: Secret
metadata:
  name: ingress-ca
type: kubernetes.io/tls
```

7. Copy the following files from your `2dcdx-active-yamls` directory to the `2dcdx-ws-yamls` directory in your warm-standby data center.

```
ca-issuer-secret.yaml
mgmt-enc-key.txt
ptl-enc-key.txt
```

These files are required to ensure that the ingress-ca and encryption secrets on both sites are the same.

Tip: To save time during the preparation of your warm-standby data center you can also copy these yaml files:

```
ptl-tls-client-cert.yaml
mgmt-tls-client-cert.yaml
```

8. If you are installing the Portal subsystem in a different namespace to the Management subsystem on the active data center, use the `ca-issuer-secret.yaml` file to create the issuer in your portal namespace. Follow the 'Before you begin' steps here: [Create issuer](#). Do not proceed to install the Portal subsystem after creating the issuer.

What to do next

Prepare your warm-standby data center [Preparing your warm-standby data center](#).

Preparing your warm-standby data center

Create the secrets, certificates, and issuers that are needed for your warm-standby data center to replicate with the active.

About this task

All operations are done on the CLI, in the `2dcdx-ws-yamls` directory you created in [Planning and initial preparation](#).

In the yaml files and commands that are shown here, replace `<namespace>` with the name of the corresponding subsystem namespace.

Procedure

1. Deploy the cert-manager.

- a. Create a file that is called `cert-manager.yaml` and paste in the following contents:

```
apiVersion: operator.ibm.com/v1alpha1
kind: OperandRequest
metadata:
  name: ibm-apiconnect
spec:
  requests:
  - operands:
    - name: ibm-cert-manager-operator
      registry: common-service
      registryNamespace: ibm-common-services
```

- b. Apply this yaml file with:

```
oc apply -f cert-manager.yaml -n <namespace>
```

- c. Confirm that the cert-manager is deployed:

```
oc get deploy -n ibm-common-services | grep ibm-cert-manager-operator
```

- d. Wait for the cert-manager deployment to become available:

```
oc wait --for=condition=available deployment.apps/ibm-cert-manager-operator -n ibm-common-services --timeout=900s
```

- e. Wait for the three cert-manager pods and operator to be in running state:

```
oc get pods -n ibm-common-services | grep cert
```

cert-manager-cainjector-566b5d5698-pzccn	1/1	Running	0	18m
cert-manager-controller-766ddff8c4-ftr85	1/1	Running	0	18m
cert-manager-webhook-79bb8f67b7-gvkst	1/1	Running	0	18m
ibm-cert-manager-operator-fdfd66bd4-6rrz6	1/1	Running	0	19m

2. Import the ingress-ca issuer secret.

This secret was created on your active data center here: [Preparing your active data center](#)

- a. If you did not do it as part of [Preparing your active data center](#), copy the `ca-issuer-secret.yaml` file to your warm-standby data center.

- b. Create the secret with:

```
oc apply -f ca-issuer-secret.yaml -n <namespace>
```

- c. Verify that the secret was created with:

```
oc get secret -n <namespace>
```

NAME	TYPE	DATA	AGE
ingress-ca	kubernetes.io/tls	3	19h

- d. If your Portal subsystem is in a different namespace in your warm-standby data center, then repeat the `oc apply` operation against the portal namespace.

3. Create a file that is called `warm-standby-certs.yaml` and paste in:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "selfsigning-issuer"
  }
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-issuer"
  }
spec:
  ca:
    secretName: ingress-ca
---
apiVersion: cert-manager.io/v1
kind: Certificate
```

```

metadata:
  name: portal-admin-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "portal-admin-client"
  }
spec:
  subject:
    organizations:
      - cert-manager
  commonName: portal-admin-client
  secretName: portal-admin-client
  issuerRef:
    name: ingress-issuer
  usages:
    - "client auth"
    - "signing"
    - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "management"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "portal-admin-client"
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-client-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-client-client"
  }
spec:
  subject:
    organizations:
      - cert-manager
  commonName: gateway-client-client
  secretName: gateway-client-client
  issuerRef:
    name: ingress-issuer
  usages:
    - "client auth"
    - "signing"
    - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "management"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "gateway-client-client"
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: analytics-ingestion-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "analytics-ingestion-client"
  }
spec:
  subject:
    organizations:
      - cert-manager
  commonName: analytics-ingestion-client
  secretName: analytics-ingestion-client
  issuerRef:
    name: ingress-issuer
  usages:
    - "client auth"
    - "signing"
    - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "management"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "analytics-ingestion-client"
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-service
  labels: {

```



```

    app.kubernetes.io/instance: "gatewaycluster",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-service"
  }
spec:
  subject:
    organizations:
    - cert-manager
    commonName: gateway-service
    secretName: gateway-service
    issuerRef:
      name: ingress-issuer
    usages:
    - "client auth"
    - "signing"
    - "key encipherment"
    duration: 17520h # 2 years
    renewBefore: 720h # 30 days
    privateKey:
      rotationPolicy: Always
    secretTemplate:
      labels:
        app.kubernetes.io/instance: "gatewaycluster"
        app.kubernetes.io/managed-by: "ibm-apiconnect"
        app.kubernetes.io/name: "gateway-service"
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-peering
  labels: {
    app.kubernetes.io/instance: "gatewaycluster",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-peering"
  }
spec:
  subject:
    organizations:
    - cert-manager
    commonName: gateway-peering
    secretName: gateway-peering
    issuerRef:
      name: ingress-issuer
    usages:
    - "server auth"
    - "client auth"
    - "signing"
    - "key encipherment"
    duration: 17520h # 2 years
    renewBefore: 720h # 30 days
    privateKey:
      rotationPolicy: Always
    secretTemplate:
      labels:
        app.kubernetes.io/instance: "gatewaycluster"
        app.kubernetes.io/managed-by: "ibm-apiconnect"
        app.kubernetes.io/name: "gateway-peering"

```

4. Apply this file by running against your Management subsystem namespace:

```
oc apply -f warm-standby-certs.yaml -n <namespace>
```

5. Verify that the certificates, secrets, and issuers were created with:

```

oc get secrets -n <namespace>
NAME                                TYPE                                DATA  AGE
analytics-ingestion-client          kubernetes.io/tls                  3      12s
gateway-client-client               kubernetes.io/tls                  3      21s
gateway-peering                     kubernetes.io/tls                  3      11s
gateway-service                     kubernetes.io/tls                  3      21s
ingress-ca                          kubernetes.io/tls                  3      18m
portal-admin-client                 kubernetes.io/tls                  3      17s

oc get certificates -n <namespace>
NAME                                READY  SECRET                                AGE  EXPIRATION
analytics-ingestion-client          True   analytics-ingestion-client           2m18s  2024-09-04T12:27:38Z
gateway-client-client               True   gateway-client-client                2m19s  2024-09-04T12:27:28Z
gateway-peering                     True   gateway-peering                     2m17s  2024-09-04T12:27:39Z
gateway-service                     True   gateway-service                     2m18s  2024-09-04T12:27:28Z
portal-admin-client                 True   portal-admin-client                 2m19s  2024-09-04T12:27:33Z

oc get issuers -n <namespace>
NAME                                READY  AGE
ingress-issuer                      True   18m
selfsigning-issuer                  True   2m42s

```

6. If your Portal subsystem is in a different namespace in your warm-standby data center, then to create your certificates, issuers, and secrets in the Portal subsystem namespace:

a. Create a file that is called `warm-standby-pt1-certs.yaml` and paste in:

```

#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#

```

```

# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "selfsigning-issuer"
  }
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-issuer"
  }
spec:
  ca:
    secretName: ingress-ca

```

b. Apply this file to your Portal subsystem namespace:

```
oc apply -f warm-standby-certs.yaml -n <namespace>
```

c. Verify that your issuers and secrets were created successfully:

```
oc get issuers -n <portal namespace>
NAME          READY   AGE
ingress-issuer  True    43s
selfsigning-issuer  True    43s

oc get secrets -n <portal namespace>
NAME          TYPE          DATA   AGE
ingress-ca    kubernetes.io/tls    3       4m18s
```

7. Create the encryption key secrets for the Management and Portal subsystems, by using the random string generated on the active data center.

Use the `mgmt-enc-key.txt` and `ptl-enc-key.txt` files that were copied to your warm-standby data center when you followed [Preparing your active data center](#).

a. Run the following command to create the management encryption key secret:

```
oc create secret generic mgmt-encryption-key --from-file=encryption_secret.bin=mgmt-enc-key.txt -n <management namespace>
```

b. Confirm that the secret was created successfully by running:

```
oc get secrets -n <management namespace> | grep mgmt-encryption-key

mgmt-encryption-key    Opaque          1       83s
```

c. Run the following command to create the portal encryption key secret:

```
oc create secret generic ptl-encryption-key --from-file=encryption_secret=ptl-enc-key.txt -n <portal namespace>
```

d. Confirm that the secret was created successfully by running:

```
oc get secrets -n <portal namespace> | grep ptl-encryption-key

ptl-encryption-key    Opaque          1       15s
```

8. Create the TLS client replication certificates for Management and Portal.

a. Create a yaml file that is called `mgmt-tls-client-cert.yaml` and paste in the following contents:

```

apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: mgmt-replication-client
spec:
  commonName: mgmt-replication-client
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: mgmt-replication-client

```

b. Apply this yaml file with:

```
oc apply -f mgmt-tls-client-cert.yaml -n <management namespace>
```

c. Verify that the certificate was created with:

```
oc get certs -n <management namespace>
```

NAME	READY	SECRET	AGE	EXPIRATION
...				
mgmt-replication-client	True	mgmt-replication-client	16m	2024-08-17T13:04:27Z

d. Create a file `ptl-tls-client-cert.yaml` and paste in the following contents:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ptl-replication-client
spec:
  commonName: ptl-replication-client
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: ptl-replication-client
```

e. Apply this yaml file with:

```
oc apply -f ptl-tls-client-cert.yaml -n <portal namespace>
```

f. Verify that the certificate was created with:

```
oc get certs -n <portal namespace>
```

NAME	READY	SECRET	AGE	EXPIRATION
...				
ptl-replication-client	True	ptl-replication-client	16m	2024-08-17T13:04:27Z

What to do next

Install API Connect on your active data center [Installing API Connect on the active data center](#).

Installing API Connect on the active data center

Add the `multiSiteHA` configuration to the API Connect Management and Portal CR YAML files before they are applied.

Before you begin

Verify that all the secrets, certificates, and issuers are ready:

```
oc get secrets -n <namespace>
ingress-ca          kubernetes.io/tls          3          3d21h
mgmt-replication-client kubernetes.io/tls          3          3d21h
ptl-replication-client kubernetes.io/tls          3          3d21h

oc get certs -n <namespace>
ingress-ca          True    ingress-ca          3d21h    2032-08-15T13:01:47Z
mgmt-replication-client True    mgmt-replication-client 3d21h    2024-08-17T13:04:27Z
ptl-replication-client True    ptl-replication-client  3d21h    2024-08-17T13:04:26Z

oc get issuer -n <namespace>
ingress-issuer      True    3d21h
self-signed         True    3d21h
```

Where `<namespace>` are the namespaces you created for API Connect.

Note: If you have separate namespaces for Management and Portal subsystems, then run the check in both namespaces. You should not see the `mgmt-replication-client` in the portal namespace, nor the `ptl-replication-client` in the management namespace.

Procedure

1. Follow these steps for installing the Management subsystem on OpenShift [Installing the Management subsystem in a shared namespace](#), but add the `encryptionSecret` and `multiSiteHA` section to the `mgmt_cr.yaml` file:

```
siteName: <dcl site name>
encryptionSecret:
  secretName: mgmt-encryption-key
multiSiteHA:
  mode: active
  replicationEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
  hosts:
    - name: mgmt-replication.<active data center ingress domain>
      secretName: mgmt-replication-server
```

```

replicationPeerFQDN: mgmt-replication.<warm-standby data center ingress domain>
tlsClient:
  secretName: mgmt-replication-client

```

where

- `<dc1 site name>` is the site name for this data center, as decided in [Planning and initial preparation](#).
- `<active data center ingress domain>` and `<warm-standby data center ingress domain>` can be determined by running the command `oc get ingresses.config/cluster -o jsonpath={.spec.domain}` in each data center.

Note: If you are installing the Management subsystem in a separate namespace, these same steps apply.

2. Apply the update `mgmt_cr.yaml` file as directed in [Installing the Management subsystem in a shared namespace](#). You can monitor the deployment with:

```

oc get ManagementCluster -n <management namespace>

```

NAME	READY	STATUS	VERSION	RECONCILED VERSION	AGE
management	19/19	Running	10.0.5.0	10.0.5.0-1281	74m

3. Follow these steps for installing the Portal subsystem on OpenShift [Installing the Portal subsystem in a shared namespace](#), but add the `encryptionSecret` and `multiSiteHA` section to the `portal_cr.yaml` file.

```

siteName: <dc1 site name>
encryptionSecret:
  secretName: ptl-encryption-key
multiSiteHA:
  mode: active
  replicationEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: ptl-replication.<active data center ingress domain>
        secretName: ptl-replication-server
  replicationPeerFQDN: ptl-replication.<warm-standby data center ingress domain>
tlsClient:
  secretName: ptl-replication-client

```

where

- `<dc1 site name>` is the site name for this data center, as decided in [Planning and initial preparation](#).
- `<active data center ingress domain>` and `<warm-standby data center ingress domain>` can be determined by running the command `oc get ingresses.config/cluster -o jsonpath={.spec.domain}` in each data center.

Note: If you are installing the Portal subsystem in a separate namespace, these same steps apply.

4. Apply the updated `portal_cr.yaml` file as directed in [Installing the Portal subsystem in a shared namespace](#).

```

oc get PortalCluster -n <portal namespace>

```

NAME	READY	STATUS	VERSION	RECONCILED VERSION	AGE
portal	6/6	Running	10.0.5.0	10.0.5.0-1281	49m

Results

Confirm that the management subsystem is ready, but in **Warning** state with `oc get mgmt:`

```
oc get mgmt -n <namespace>
```

NAME	READY	STATUS	VERSION	RECONCILED VERSION	MESSAGE
management	n/n	Warning	10.0.5.3-0	10.0.5.3-0	Management is ready. HA Status Warning - see HAStatus in CR for details
	8m59s				

```

status:
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication not working, install or upgrade in progress.",
    "reason": "na",
    "status": "True",
    "type": "Pending"
  }

```

The management CR is expected to report the status of **Warning** until the warm-standby management subsystem is deployed, and both management subsystems complete data replication. When you see the status message "Management is ready. HA Status Warning - see HAStatus in CR for details", you can move on to [Installing API Connect on the warm-standby data center](#).

What to do next

Install API Connect in your warm-standby data center [Installing API Connect on the warm-standby data center](#).

Deploy your Gateway and Analytics subsystems in your active data center. For shared namespace: [Installing with subsystem CRs in a shared namespace on OpenShift](#) or multiple namespaces: [Installing with subsystem CRs in different namespaces or environments on OpenShift](#).

Installing API Connect on the warm-standby data center

Add the `multiSiteHA` configuration to the API Connect Management and Portal YAML files before they are applied.

Before you begin

Verify that all the secrets, certificates, and issuers are ready:

```
oc get secrets -n <namespace>
ingress-ca          kubernetes.io/tls          3          3d21h
mgmt-replication-client kubernetes.io/tls          3          3d21h
ptl-replication-client kubernetes.io/tls          3          3d21h

oc get certs -n <namespace>
mgmt-replication-client True      mgmt-replication-client 3d21h  2024-08-17T13:04:27Z
ptl-replication-client True      ptl-replication-client  3d21h  2024-08-17T13:04:26Z

oc get issuer -n <namespace>
ingress-issuer      True      3d21h
self-signed         True      3d21h
```

Where `<namespace>` are the namespaces you created for API Connect.

Note: If you have separate namespaces for Management and Portal subsystems, then run the check in both namespaces. You should not see the `mgmt-replication-client` in the portal namespace, nor the `ptl-replication-client` in the management namespace.

Procedure

1. Follow these steps for installing the Management subsystem on OpenShift [Installing the Management subsystem in a shared namespace](#), but add the `encryptionSecret` and `multiSiteHA` section to the `mgmt_cr.yaml` file:

```
siteName: <dc2 site name>
encryptionSecret:
  secretName: mgmt-encryption-key
multiSiteHA:
  mode: passive
  replicationEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
  hosts:
    - name: mgmt-replication.<warm-standby data center ingress domain>
      secretName: mgmt-replication-server
  replicationPeerFQDN: mgmt-replication.<active data center ingress domain>
tlsClient:
  secretName: mgmt-replication-client
```

where

- `<dc2 site name>` is the site name for this data center, as decided in [Planning and initial preparation](#).
- `<active data center ingress domain>` and `<warm-standby data center ingress domain>` can be determined by running the command `oc get ingresses.config/cluster -o jsonpath={.spec.domain}` in each data center.

Note: If you are installing the Management subsystem in a separate namespace, these same steps apply.

Remember: Make sure that the management endpoints are set the same as for your active data center deployment.

2. Apply the update `mgmt_cr.yaml` file as directed in [Installing the Management subsystem in a shared namespace](#). Monitor the deployment, but note that on the warm-standby there are fewer pods than on the active:

```
oc get ManagementCluster -n <namespace>NAME          READY   STATUS    VERSION   RECONCILED VERSION   AGE
management  n/n     Running  10.0.5.3  10.0.5.3-1281  62m
```

3. Follow these steps for installing the Portal subsystem on OpenShift [Installing the Portal subsystem in a shared namespace](#), but add the `encryptionSecret` and `multiSiteHA` section to the `portal_cr.yaml` file.

```
siteName: <dc2 site name>
encryptionSecret:
  secretName: ptl-encryption-key
multiSiteHA:
  mode: passive
  replicationEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
  hosts:
    - name: ptl-replication.<warm-standby data center ingress domain>
      secretName: ptl-replication-server
  replicationPeerFQDN: ptl-replication.<warm-standby data center ingress domain>
tlsClient:
  secretName: ptl-replication-client
```

where

- `<dc2 site name>` is the site name for this data center, as decided in [Planning and initial preparation](#).
- `<active data center ingress domain>` and `<warm-standby data center ingress domain>` can be determined by running the command `oc get ingresses.config/cluster -o jsonpath={.spec.domain}` in each data center.

Remember: Make sure that the portal endpoints are set the same as for your active data center deployment.

4. Apply the updated `portal_cr.yaml` file as directed in [Installing the Portal subsystem in a shared namespace](#).

```
oc get PortalCluster -n <namespace>
NAME   READY   STATUS    VERSION   RECONCILED VERSION   AGE
portal  6/6     Running  10.0.5.3  10.0.5.3-1281  47m
```

Results

While the management subsystems on the warm-standby and active data centers are synchronizing their databases, the management status reports **Warning**, and the `haStatus` reports `pending`:

```
oc get mgmt -n <namespace>
```

NAME	READY	STATUS	VERSION	RECONCILED VERSION	MESSAGE
management for details	n/n 8m59s	Warning	10.0.5.3-0	10.0.5.3-0	Management is ready. HA Status Warning - see HAStatus in CR

```

status:
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication not working, install or upgrade in progress.",
    "reason": "na",
    "status": "True",
    "type": "Pending"
  }

```

When the management database replication between sites is complete, the management status reports `Running`, and `status.haStatus` reports `Ready`:

NAME	READY	STATUS	VERSION	RECONCILED VERSION	MESSAGE
management	n/n	Running	10.0.5.3-0	10.0.5.3-0	Management is ready. 8m59s

```

status:
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication is working",
    "reason": "na",
    "status": "True",
    "type": "Ready"
  }

```

If the management database replication between sites fails for any reason other than because an install or upgrade is in progress, the `haStatus` output shows:

NAME	READY	STATUS	VERSION	RECONCILED VERSION	MESSAGE
management for details	n/n 8m59s	Warning	10.0.5.3-0	10.0.5.3-0	Management is ready. HA Status Warning - see HAStatus in CR

```

status:
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication not working",
    "reason": "na",
    "status": "True",
    "type": "Warning"
  }

```

If the warning persists, see [Troubleshooting a two data center deployment](#).

You can validate that your portal deployments are synchronizing by running `oc get pods` on both the active and warm-standby data centers. Confirm that the number and names of the pods all match (the UUIIDs in the names might be different on each site), and that all are in the `Ready` state.

For additional replication verification checks, see [Verifying replication between data centers](#). It is recommended to run a test failover, and confirm that all of the expected data is present and correct on the newly active site. See [How to failover API Connect from the active to the warm-standby data center](#) for details.

What to do next


Deploy your Gateway and Analytics subsystems for shared namespace: [Installing with subsystem CRs in a shared namespace on OpenShift](#) or multiple namespaces: [Installing with subsystem CRs in different namespaces or environments on OpenShift](#)

Configure your deployment: [Cloud Manager configuration checklist](#), and complete a test failover to verify that your data replicated to the warm-standby. See [How to failover API Connect from the active to the warm-standby data center](#) for details.

Troubleshooting installation and upgrade on OpenShift

Review the following troubleshooting tips if you encounter a problem while installing or upgrading API Connect on OpenShift, including as a component of IBM Cloud Pak for Integration (CP4I).

- [One or more pods in CrashLoopBackoff or Error state, and report a certificate error in the logs](#)
- [Pgbouncer pod enters a CrashLoopBackoff state when upgrading to API Connect 10.0.5.3](#)
- [You see the denied: insufficient scope error during an air-gapped deployment](#)
- [Apiconnect operator crashes](#)
- [Upgrading a 3 node profile to IBM API Connect 10.0.3.0 might result in some portal-db/www pods being stuck in the Pending state](#)
- [Issues when installing Drupal 8 based custom modules or sub-themes into the Drupal 9 based Developer Portal](#)
- [Disabling the Portal web endpoint check](#)

Note: In the  Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.

One or more pods in `CrashLoopBackoff` or `Error` state, and report a certificate error in the logs

In rare cases, cert-manager might detect a certificate in a bad state right after it has been issued, and then re-issues the certificate. If a CA certificate has been issued twice, the certificate that was signed by the previously issued CA will be left stale and can't be validated by the newly issued CA. In this scenario, one of the following messages displays in the log:

- `javax.net.ssl.SSLHandshakeException: Received fatal alert: certificate_unknown`
- `Error: unable to verify the first certificate`
- `ERROR: openssl verify failed to verify the Portal CA tls.crt, ca.crt chain signed the Portal Server tls.crt cert`

Resolve the problem by completing the following steps:

1. Use `apicops` (v10 version 0.10.57+ required) to validate the certificates in the system:

```
apicops upgrade:stale-certs -n <namespace>
```

2. If any certificate that is managed by cert-manager fails the validation, delete the stale certificate secret:

```
oc delete secret <stale-secret> -n <namespace>
```

Cert-manager automatically generates a new certificate to replace the one you deleted.

3. Use `apicops` to make sure all certificates can be verified successfully:

```
apicops upgrade:stale-certs -n <namespace>
```

Pgbouncer pod enters a `CrashLoopBackoff` state when upgrading to API Connect 10.0.5.3

This issue might be caused by a corrupt compliance entry in the `pgbouncer.ini` file. When the operator is upgraded to v10.0.5.3 but the `ManagementCluster` CR is not yet upgraded, the operator might update the `pgbouncer.ini` file in the `PGBouncer` secret with the older `ManagementCluster` CR's profile file, which does not contain any value for the `compliance pool_size`. As a result, the value gets incorrectly set to the string `<no value>`.

In the majority of cases this bad configuration update is temporary, until the `ManagementCluster` CR's version is also updated as part of the upgrade process and the problem is resolved. The issue will only become evident if the `pgbouncer` pod restarts prior to the `ManagementCluster` CR's version field being updated.

After the operator updates, if the other pods are restarted (specifically the `pgbouncer` pod) before the `ManagementCluster` CR's version is updated, then the `pgbouncer` pod will get stuck in a `CrashLoopBackOff` state due to the missing value for the `compliance pool_size` setting.

The following example shows what the error looks like in the `pgbouncer` logs:

```
Wed May 17 09:00:02 UTC 2023 INFO: Starting pgBouncer..
2023-05-17 09:00:02.897 UTC [24] ERROR syntax error in connection string
2023-05-17 09:00:02.897 UTC [24] ERROR invalid value "host=mgmt-a516d013-postgres port=5432 dbname=compliance pool_size=<no value>" for parameter compliance in configuration (/pgconf/pgbouncer.ini:7)
2023-05-17 09:00:02.897 UTC [24] FATAL cannot load config file
```

Resolve the issue by running the commands in the following script, where `<namespace>` is the API Connect instance's namespace if you deployed using the top-level `APIConnectCluster` CR, or is the management subsystem's namespace if you deploying using individual subsystem CRs.

```
NAMESPACE=<APIC-namespace>
BOUNCER_SECRET=<management-prefix>-postgres-pgbouncer-secret
TEMP_FILE=/tmp/pgbouncer.ini

**Step 1 - Common: Get the existing pgbouncer.ini file**
oc get secret -n $NAMESPACE $BOUNCER_SECRET -o jsonpath='{.data.pgbouncer\.ini}' | base64 -d > $TEMP_FILE

**Step 2 - Linux version: Update the file and use it to patch the Secret on the cluster**
sed 's/<no value>/20/' $TEMP_FILE | base64 -w0 | xargs -I{} kubectl patch secret -n $NAMESPACE $BOUNCER_SECRET --type='json' -p="{['op': 'replace', 'path': '/data/pgbouncer.ini', 'value': {} ]}"

**Step 2 - Mac version: Update the file and use it to patch the Secret on the cluster**
sed 's/<no value>/20/' $TEMP_FILE | base64 -b0 | xargs -S2000 -I{} kubectl patch secret -n $NAMESPACE $BOUNCER_SECRET --type='json' -p="{['op': 'replace', 'path': '/data/pgbouncer.ini', 'value': {} ]}"

# Step 3 - Common: Restart pgbouncer to pick up the updated Secret configuration
oc delete pod <bouncer_pod_name> -n $NAMESPACE
```

You see the denied: `insufficient scope` error during an air-gapped deployment

Problem: You encounter the `denied: insufficient scope` message while mirroring images during an air-gapped installation or upgrade.

Reason: This error occurs when a problem is encountered with the entitlement key used for obtaining images.

Solution: Obtain a new entitlement key by completing the following steps:

1. Log in to the [IBM Container Library](#).
2. In the Container software library, select Get entitlement key.
3. After the Access your container software heading, click Copy key.
4. Copy the key to a safe location.

Apiconnect operator crashes

Problem: During installation (or upgrade), the Apiconnect operator crashes with the following message:

```
panic: unable to build API support: unable to get Group and Resources: unable to retrieve the complete list of server APIs:
packages.operators.coreos.com/v1: the server is currently unable to handle the request
```

```
goroutine 1 [running]:
github.ibm.com/velox/apiconnect-operator/operator-utils/v2/apiversions.GetAPISupport(0x0)
    operator-utils/v2/apiversions/api-versions.go:89 +0x1e5
main.main()
    ibm-apiconnect/cmd/manager/main.go:188 +0x4ee
```

Additional symptoms:

- Apiconnect operator is in crash loopback status
- Kube apiserver pods log the following information:

```
E1122 18:02:07.853093 18 available_controller.go:437] v1.packages.operators.coreos.com failed with:
failing or missing response from https://10.128.0.3:5443/apis/packages.operators.coreos.com/v1:
bad status from https://10.128.0.3:5443/apis/packages.operators.coreos.com/v1: 401
```

- The IP logged here belongs to the *package server* pod present in the *openshift-operator-lifecycle-manager* namespace
- Package server pods log the following: */apis/packages.operators.coreos.com/v1* API call is being rejected with 401 issue

```
E1122 18:10:25.614179 1 authentication.go:53] Unable to authenticate the request due to an error: x509:
certificate signed by unknown authority I1122 18:10:25.614224 1 httplog.go:90]
verb="GET" URI="/apis/packages.operators.coreos.com/v1" latency=161.243µs resp=401
UserAgent="Go-http-client/2.0" srcIP="10.128.0.1:41370":
```

- Problem is intermittent

Solution:

- If you find the exact symptoms as described, the solution is to delete package server pods in the *openshift-operator-lifecycle-manager* namespace.
- New package server pods will log the **200 Success** message for the same API call.

Upgrading a 3 node profile to IBM API Connect 10.0.3.0 might result in some *portal-db/www* pods being stuck in the *Pending* state

IBM® API Connect 10.0.3.0 introduces the pod anti-affinity required rule, meaning that in a 3 node profile deployment, all 3 db and www pods can run only if there are at least 3 running worker nodes. This rule can cause some upgrades to version 10.0.3.0 to become stuck in the *Pending* state, in which case some extra steps are needed during the upgrade to workaround the issue. See the following example for detailed information about the issue, and how to continue with the upgrade.

Important: You must have a backup of your current deployment before starting the upgrade.

The following example shows 3 worker nodes:

```
$ oc get nodes
NAME                STATUS    ROLES    AGE   VERSION
apimdev0103         Ready     worker   42m   v1.20.0
apimdev0129         Ready     worker   45m   v1.20.0
apimdev1066         Ready     worker   39m   v1.20.0
```

The pods have been scheduled across only 2 of the 3 worker nodes due to a transient problem with *apimdev1066*, as shown in the following pod list. Pods without persistent storage, such as *nginx-X*, can be rescheduled to *apimdev1066* as soon as they are restarted, but any pods with persistent local storage, such as *db-X* and *www-X*, have to be rescheduled onto the same worker node as that is where their files live.

```
$ oc get po -o wide
NAME                READY    STATUS    RESTARTS   AGE   IP              NODE                NOMINATED NODE
READINESS GATES
ejs-portal-nginx-84f57ffd8c-hbf66 1/1      Running   0           5m12s  888.16.109.208  apimdev0103         <none>         <none>
ejs-portal-nginx-84f57ffd8c-mvq96 1/1      Running   0           5m12s  888.16.142.215  apimdev0129         <none>         <none>
ejs-portal-nginx-84f57ffd8c-vpmt1 1/1      Running   0           5m12s  888.16.142.214  apimdev0129         <none>         <none>
ejs-portal-sitel-db-0              2/2      Running   0           4m39s  888.16.109.209  apimdev0103         <none>         <none>
ejs-portal-sitel-db-1              2/2      Running   0           6m37s  888.16.109.206  apimdev0103         <none>         <none>
ejs-portal-sitel-db-2              2/2      Running   0           4m39s  888.16.142.216  apimdev0129         <none>         <none>
ejs-portal-sitel-www-0             2/2      Running   0           4m9s   888.16.109.210  apimdev0103         <none>         <none>
ejs-portal-sitel-www-1             2/2      Running   0           6m37s  888.16.142.213  apimdev0129         <none>         <none>
ejs-portal-sitel-www-2             2/2      Running   0           4m9s   888.16.142.217  apimdev0129         <none>         <none>
ibm-apiconnect-75b47f9f87-p25dd    1/1      Running   0           5m12s  888.16.109.207  apimdev0103         <none>         <none>
```

The upgrade to version 10.0.3.0 is started and the following pod list is observed:

```
$ oc get po -o wide
NAME                READY    STATUS    RESTARTS   AGE   IP              NODE                NOMINATED NODE
READINESS GATES
ejs-portal-nginx-84f57ffd8c-hbf66 1/1      Running   0           10m    888.16.109.208  apimdev0103         <none>         <none>
ejs-portal-nginx-84f57ffd8c-mvq96 1/1      Running   0           10m    888.16.142.215  apimdev0129         <none>         <none>
ejs-portal-nginx-84f57ffd8c-vpmt1 1/1      Running   0           10m    888.16.142.214  apimdev0129         <none>         <none>
ejs-portal-sitel-db-0              2/2      Running   0           10m    888.16.109.209  apimdev0103         <none>         <none>
ejs-portal-sitel-db-1              0/2      Pending   0           91s    <none>          <none>         <none>
ejs-portal-sitel-db-2              2/2      Running   0           2m41s  888.16.142.218  apimdev0129         <none>         <none>
ejs-portal-sitel-www-0             2/2      Running   0           9m51s  888.16.109.210  apimdev0103         <none>         <none>
ejs-portal-sitel-www-1             2/2      Running   0           12m    888.16.142.213  apimdev0129         <none>         <none>
ejs-portal-sitel-www-2             0/2      Pending   0           111s   <none>         <none>         <none>
ibm-apiconnect-75b47f9f87-p25dd    1/1      Running   0           10m    888.16.109.207  apimdev0103         <none>         <none>
```

The pod list shows that *db-2* has restarted, and has been rescheduled to *apimdev0129* as there were no other db pods running on that node. However, *db-1* and *www-2* are both stuck in *Pending* state, as there is already a pod of the same type running on the worker node that is hosting the local storage that they are bound to. If you run a *describe* command of either pod you will see the following output:

```
Events:
Type      Reason          Age          From          Message
-----
Warning   FailedScheduling 10s (x4 over 2m59s)  default-scheduler  0/3 nodes are available: 1 node(s) didn't match pod affinity/anti-affinity, 1 node(s) didn't match pod anti-affinity rules, 2 node(s) had volume node affinity conflict.
```


To resolve this situation you need to delete the PVCs for each pod, and then delete the pod itself, so that Kubernetes will regenerate the PVCs and schedule the pod on the worker node that does not have the anti-affinity conflict.

Therefore, for the `db-1` pod the following commands must be run:

```
$ oc get pvc | grep ejs-portal-sitel-db-1
db-ejs-portal-sitel-db-1      Bound      local-pv-fa445e30    250Gi      RWO          local-storage    15m
dblogs-ejs-portal-sitel-db-1  Bound      local-pv-d57910e7    250Gi      RWO          local-storage    15m

$ oc delete pvc db-ejs-portal-sitel-db-1 dblogs-ejs-portal-sitel-db-1
persistentvolumeclaim "db-ejs-portal-sitel-db-1" deleted
persistentvolumeclaim "dblogs-ejs-portal-sitel-db-1" deleted

$ oc delete po ejs-portal-sitel-db-1
pod "ejs-portal-sitel-db-1" deleted
```

For the `www-2` pod the following commands must be run:

```
$ oc get pvc | grep ejs-portal-sitel-www-2
admin-ejs-portal-sitel-www-2  Bound      local-pv-48799536    245Gi      RWO          local-storage    51m
backup-ejs-portal-sitel-www-2  Bound      local-pv-a93f5607    245Gi      RWO          local-storage    51m
web-ejs-portal-sitel-www-2     Bound      local-pv-facd4489    245Gi      RWO          local-storage    51m

$ oc delete pvc admin-ejs-portal-sitel-www-2 backup-ejs-portal-sitel-www-2 web-ejs-portal-sitel-www-2
persistentvolumeclaim "admin-ejs-portal-sitel-www-2" deleted
persistentvolumeclaim "backup-ejs-portal-sitel-www-2" deleted
persistentvolumeclaim "web-ejs-portal-sitel-www-2" deleted

$ oc delete po ejs-portal-sitel-www-2
pod "ejs-portal-sitel-www-2" deleted
```

If the PVC has `persistentVolumeReclaimPolicy: Delete` set on it, no cleanup is necessary, as the old data will have been deleted on the worker node that is no longer running the `db-1` and `www-2` pods. However, if you are using `local-storage` in a Kubernetes installation, then there might be some cleanup to do to remove the old data from the worker node that had the anti-affinity conflict.

Kubernetes can now reschedule the pods. All pods with persistent storage are now spread across the available worker nodes, and the pods whose PVCs were deleted will get a full copy of the data from the existing running pods. The following pod list is now observed in our example:

```
$ oc get po -o wide
NAME                                     READY   STATUS    RESTARTS   AGE   IP              NODE                   NOMINATED NODE
READINESS GATES
ejs-portal-nginx-84f57ffd8c-f85wm      1/1     Running  0          30s   888.16.29.136  apimdev1066           <none>          <none>
ejs-portal-nginx-84f57ffd8c-k5klb      1/1     Running  0          103s  888.16.142.220  apimdev0129           <none>          <none>
ejs-portal-nginx-84f57ffd8c-lhqqs      1/1     Running  0          1m53s 888.16.109.212  apimdev0103           <none>          <none>
ejs-portal-sitel-db-0                  2/2     Running  0          6m43s 888.16.109.211  apimdev0103           <none>          <none>
ejs-portal-sitel-db-1                  2/2     Running  0          8m20s 888.16.29.134  apimdev1066           <none>          <none>
ejs-portal-sitel-db-2                  2/2     Running  0          14m   888.16.142.218  apimdev0129           <none>          <none>
ejs-portal-sitel-www-0                 2/2     Running  0          93s   888.16.109.213  apimdev0103           <none>          <none>
ejs-portal-sitel-www-1                 2/2     Running  0          3m55s 888.16.142.219  apimdev0129           <none>          <none>
ejs-portal-sitel-www-2                 2/2     Running  0          7m27s 888.16.29.135  apimdev1066           <none>          <none>
ibm-apiconnect-75b47f9f87-p25dd       1/1     Running  0          22m   888.16.109.207  apimdev0103           <none>          <none>
```

Issues when installing Drupal 8 based custom modules or sub-themes into the Drupal 9 based Developer Portal

From IBM API Connect 10.0.3.0, the Developer Portal is based on the Drupal 9 content management system. If you want to install Drupal 8 custom modules or sub-themes into the Drupal 9 based Developer Portal, you must ensure that they are compatible with Drupal 9, including any custom code that they contain, and not using any deprecated APIs, for example. There are tools available for checking your custom code, such as [drupal check](#) on GitHub, which checks Drupal code for deprecations.

For example, any Developer Portal sites that contain modules or sub-themes that don't contain a Drupal 9 version declaration will fail to upgrade, and errors like the following output will be seen in the `admin` logs:

```
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: Checking theme: emeraldgreen
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '' found for emeraldgreen
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: Checking theme: rubyred
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '8.x' found for rubyred
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: Found themes incompatible with
Drupal 9: emeraldgreen rubyred
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: /tmp/restore_site.355ec8 is NOT
Drupal 9 compatible
...
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: Checking module: custom_mod_1
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '' found for custom_mod_1
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: Checking module: custom_mod_2
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '8.x' found for custom_mod_2
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: Found modules incompatible with
Drupal 9: emeraldgreen rubyred
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: sitel.com is NOT Drupal 9
compatible
```

To fix version compatibility errors, all custom modules and sub-themes should declare a `core_version_requirement` key in their `*.info.yml` file that indicates Drupal 9 compatibility. For example:

```
name: Example module
type: module
description: Purely an example
core: 8.x
```

```
core_version_requirement: '^8 || ^9'  
package: Example module
```

```
# Information added by Drupal.org packaging script on 2020-05-31  
version: '8.x-1.3'  
project: 'example_module'  
datestamp: 1590905415
```

This example specifies that the module is compatible with all versions of Drupal 8 and 9. For more information, see [Let Drupal know about your module with an .info.yml file](#) on the drupal.org website.

If you have a backup of a site that you need to restore, and are getting the version compatibility error, but the module or theme *.info.yml file cannot be changed easily, then you have two options. Either:

- Add an environment variable into the portal CR for the **www** pod of the **admin** container stating **SKIP_D9_COMPAT_CHECK: "true"**. However, if you choose this method, you must be positive that all of the custom modules and themes for your sites are Drupal 9 compatible, as otherwise the sites may end up inaccessible after the upgrade or restore. On OpenShift or IBM Cloud Pak for Integration, edit the top level CR by using the UI, and add the environment variable as shown in the following CR snippet:

```
spec:  
  ...  
  portal:  
    template:  
      - containers:  
        - env:  
          - name: SKIP_D9_COMPAT_CHECK  
            value: "true"  
          name: admin  
        name: www
```

Or:

- Extract the site backup, edit the relevant files inside it, and then tar the backup file again. Note that this procedure will overwrite the original backup file, so ensure that you keep a separate copy of the original file before you start the extraction. For example:
 1. `mkdir /tmp/backup`
 2. `cd /tmp/backup`
 3. `tar xfz path_to_backup.tar.gz`
 4. Edit the custom module and theme files to make them Drupal 9 compatible, and add the correct `core_version_requirement` setting.
 5. `rm -f path_to_backup.tar.gz`
 6. `tar cfz path_to_backup.tar.gz`
 7. `cd /`
 8. `rm -rf /tmp/backup`

Disabling the Portal web endpoint check

When you create or register a Developer Portal service, the Portal subsystem checks that the Portal web endpoint is accessible. However sometimes, for example due to the complexity of public and private networks, the endpoint cannot be reached. The following example shows the errors that you might see in the `portal-www` pod, admin container logs, if the endpoint cannot be reached:

```
An error occurred contacting the provided portal web endpoint: example.com  
The provided Portal web endpoint example.com returned HTTP status code 504
```

In this instance, you can disable the Portal web endpoint check so that the Developer Portal service can be created successfully. To disable the endpoint check, complete the following update:

On Kubernetes, OpenShift, and IBM Cloud Pak for Integration
Add the following section to the Portal custom resource (CR) template:

```
spec:  
  template:  
    - containers:  
      - env:  
        - name: PORTAL_SKIP_WEB_ENDPOINT_VALIDATION  
          value: "true"  
        name: admin  
      name: www
```

Upgrading API Connect

You can upgrade a v10 deployment of API Connect to a newer version.

About this task

Select from the following sections to find detailed information about upgrading your API Connect deployment.

Attention: Upgrading to v10.0.5.x from v10.0.4-ifix3 or later, or v10.0.1.7 or later, will result in the deletion of existing analytics data. If you want to retain your analytics data then you must export it before upgrade. For instructions on exporting see [Additional considerations for upgrading analytics from a version prior to 10.0.5.0](#).

Note: For information about how to maintain a two data center deployment, including upgrade considerations, see [Maintaining a two data center deployment](#).

- [Upgrading on Kubernetes](#)
Upgrade API Connect on native Kubernetes distributions.
- [Upgrading on OpenShift and Cloud Pak for Integration](#)
You can upgrade API Connect in an OpenShift environment and as part of IBM Cloud Pak for Integration.
- [Upgrading a two data center deployment on Kubernetes and OpenShift](#)
How to upgrade a two data center disaster recovery (DR) deployment on Kubernetes and OpenShift.

- [Upgrading a two data center deployment on Cloud Pak for Integration](#)
How to upgrade a two data center disaster recovery (DR) deployment on Cloud Pak for Integration.

Upgrading on Kubernetes

Upgrade API Connect on native Kubernetes distributions.

After reviewing the overview, use the procedures in this section to upgrade API Connect on native Kubernetes distributions.

Note: For upgrading from v5, see [Migrating a Version 5 deployment](#).

- [Upgrade considerations on native Kubernetes](#)
Review the supported versions, requirements, and limitations for upgrading API Connect on Kubernetes.
- [Upgrading subsystems on native Kubernetes](#)
Upgrade your API Connect subsystems to the latest version on native Kubernetes.
- [Upgrading from 2018 on native Kubernetes](#)
You can upgrade from the latest release of v2018 to the latest release of v10.
- [Troubleshooting upgrades on Kubernetes](#)
Review the following troubleshooting guidance if you encounter a problem while upgrading API Connect on Kubernetes.
- [Migrating API test data](#)
Back up API testing data before upgrading your IBM® API Connect to V10.0.1, and then restore the data after the upgrade is complete.

Upgrade considerations on native Kubernetes

Review the supported versions, requirements, and limitations for upgrading API Connect on Kubernetes.

Contents:

- [Supported upgrade paths](#)
- [Ensuring compatibility with Drupal 10 for Developer Portal customizations](#)
- [Exporting analytics data before upgrading from a version prior to 10.0.5.0](#)
- [Validating an API definition file with the API Dev Tools Swagger Parser](#)
- [x-ibm-gateway-cors-allow-credentials-when-no-cors-policy API property is deprecated](#)

Tip: After upgrade, clear your browser cache, and open a new browser window. This action avoids stale cache issues in your browser, that can result in unexpected behavior in the Cloud Manager and API Manager UIs.

Supported upgrade paths

Attention: API Connect is not supported on a FIPS-enabled environment.

Table 1. Supported upgrade paths on Kubernetes

Upgrade from:	How to upgrade to 10.0.5.4
<ul style="list-style-type: none"> • 10.0.5.3 • 10.0.5.2 • 10.0.5.1 • 10.0.5.0 • 10.0.4.0- ifix3 • 10.0.1.12- eus • 10.0.1.11- eus • 10.0.1.9- eus • 10.0.1.8- eus • 10.0.1.7- eus 	Complete the procedure in Upgrading subsystems on native Kubernetes in this 10.0.5 documentation..
Older versions of 10.0.x: <ul style="list-style-type: none"> • 10.0.4.0- ifix2 • 10.0.4.0- ifix1 • 10.0.4.0 • 10.0.3.0- ifix1 • 10.0.3.0 • 10.0.2.0 	Complete 2 upgrades: <ol style="list-style-type: none"> 1. First, upgrade to 10.0.4.0-ifix3 as explained in Upgrading subsystems on native Kubernetes in the API Connect 10.0.4 documentation. 2. Then, upgrade to 10.0.5.4 as explained in Upgrading subsystems on native Kubernetes in this documentation.

Upgrade from:	How to upgrade to 10.0.5.4
Older versions of 10.0.1.x: <ul style="list-style-type: none"> 10.0.1.6-eus 10.0.1.5-ifix4-eus 10.0.1.5-ifix3-eus 10.0.1.5-eus 10.0.1.4-ifix1-eus 10.0.1.4-eus 10.0.1.2-ifix2-eus 10.0.1.2-ifix1-eus 10.0.1.2-eus 	Complete 2 upgrades: <ol style="list-style-type: none"> First, upgrade to 10.0.1.8-eus as explained in Upgrading to 10.0.1.8-eus on Kubernetes in the API Connect 10.0.1.x-eus documentation. You do not need to upgrade to a higher version because you can move directly from 10.0.1.8-eus to 10.0.5.4. Then, upgrade to 10.0.5.4 as explained in Upgrading subsystems on native Kubernetes in this documentation.
<ul style="list-style-type: none"> 10.0.1.1 10.0.1.0 10.0.0.0-ifix2 	Complete 3 upgrades: <ol style="list-style-type: none"> First, upgrade to 10.0.1.2-ifix1-eus as explained in Upgrading from v10 to v10.0.1.2 on Kubernetes in the API Connect 10.0.1.x-eus documentation. Next, upgrade to 10.0.1.8-eus as explained in Upgrading to 10.0.1.8-eus on Kubernetes in the API Connect 10.0.1.x-eus documentation. You do not need to upgrade to a higher version because you can move directly from 10.0.1.8-eus to 10.0.5.4. Then, upgrade to 10.0.5.4 as explained in Upgrading subsystems on native Kubernetes in this documentation.
2018 FP24 and higher	Complete the procedure in Upgrading from 2018 on native Kubernetes in this documentation.

Attention: Different releases of API Connect support different versions of Kubernetes, so you might need to upgrade Kubernetes before upgrading API Connect. For more information, see [IBM API Connect Version 10 software product compatibility requirements](#).

Ensuring compatibility with Drupal 10 for Developer Portal customizations

Review the [Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10](#) to ensure that any customizations to the Developer Portal are compatible with Drupal 10.

In API Connect 10.0.5.3, the Developer Portal moved from Drupal 9 to Drupal 10 (this upgrade also requires PHP 8.1). The upgrade tooling will update your Developer Portal sites; however, if you have any custom modules or themes, it is your responsibility to ensure their compatibility with Drupal 10 and PHP 8.1 before starting the upgrade.

Note: If you are upgrading to 10.0.5.3 from 10.0.1.7 or earlier, or from 10.0.5.0 or earlier, your Developer Portal sites will first be upgraded to Drupal 9.5 and then to Drupal 10. When you are planning your the upgrade to 10.0.5.3, consider the extended time frame. To shorten your upgrade window, consider removing the Developer Portal sites that you do not need.

Exporting analytics data before upgrading from a version prior to 10.0.5.0

If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher): Upgrading to 10.0.5.x from an earlier version will result in the deletion of existing analytics data. If you want to retain your analytics data then you must export it before starting the upgrade. For instructions on exporting data, see [Additional considerations for upgrading analytics from a version prior to 10.0.5.0](#)

Validating an API definition file with the API Dev Tools Swagger Parser

From IBM API Connect Version 10.0.3.0 or later, SwaggerDoc (OpenAPI 2.0) and OpenAPI 3.0 documents that form API definition files are now validated at stage or publish time by the API Dev Tools Swagger Parser. Before upgrading, you should ensure that your existing API definition files can be parsed and are considered valid by the Swagger Parser tool. Note that any API definitions that are already staged or published are not impacted, but would undergo Swagger Parser validation again should they be re-staged or re-published in the future.

You can use either of the following techniques to ensure that existing API definition files are can be parsed and considered valid by the Swagger Parser:

- Download and install the `swagger-parser-cli` tool (see <https://www.npmjs.com/package/swagger-cli>). This tool includes a `validate` command that can validate local API definition files against the SwaggerDoc (OpenAPI 2.0) and OpenAPI 3.0 schemas, in both JSON and YAML formats.
- Use the browser-based Swagger Parser Validator available at <https://apitools.dev/swagger-parser/online/>; this accepts a URL to existing files available online, or can accept text copied and pasted into the tool, in both JSON and YAML formats. It can likewise validate against the SwaggerDoc (OpenAPI 2.0) and OpenAPI 3.0 schemas.

Note: Staging or publishing an API in API Connect results in additional syntactic validation of IBM extensions to SwaggerDoc (OpenAPI 2.0) and OpenAPI 3.0, as well as semantic validation to make sure that policies and other resources that the API refers to are valid within the context of the staging or publishing operation. Any errors associated with this additional validation will not be caught by the local or online Swagger Parser tool, nor will they be caught by the `apic validate` toolkit CLI command.

The Swagger Parser tool will report an invalid OpenAPI document. For example, if a document is missing a mandatory section such as `info` or `paths`, the following error is reported:

```
[object Object] is not a valid Swagger API definition
```

If a required property is missing from a given section, a specific error is reported. For example, if the `info` section is missing the required `version` property, the following error is reported:

```
Swagger schema validation failed.
Missing required property: version at #/info
```

`JSON_OBJECT_VALIDATION_FAILED`

Additionally, if the document contains any local references (that is, values of `$ref` properties) that do not resolve to an existing reference within the document (known as de-referencing), this is also be flagged as an error. For example, if a document contains the following as a reference:

```
schema:
  $ref: '#/definitions/Pet'
```

then the definition of `Pet` (that is, `#/definitions/Pet`) **must** exist in the same document, at the given path (`definitions` section in this example), similar to the following:

```
definitions:
  Pet:
    type: object
    required:
    - name
    properties:
      id:
        type: integer
        format: int64
```

If the local reference is not found within the document, the following de-referencing error is reported:

```
Token "Pet" does not exist.
```

x-ibm-gateway-cors-allow-credentials-when-no-cors-policy API property is deprecated

In v2018, when cross-origin resource sharing (CORS) is enabled for an API, every response to a CORS request contains the header:

```
Access-Control-Allow-Credentials: true
```

Inclusion of Access-Control-Allow-Credentials (ACAC) does not provide optimal security, so starting with v2018.4.1.17, an API property can be set to prevent this header from being returned:

```
Name: x-ibm-gateway-cors-allow-credentials-when-no-cors-policy
Collection: *
Value (string): off
```

The `Collection` property refers to a specific Catalog to which a property value applies. The "*" value means the value applies to all catalogs. You can either specify * or a Catalog (collection) name.

The property can also be set by using the DataPower CLI:

```
api-definition myAPI
  property x-ibm-gateway-cors-allow-credentials-when-no-cors-policy * off
exit
```

Important: The value of this property must be exactly the string `off` (case-sensitive) in order to remove the header.

It is only possible to turn off this header, or leave it on, for an entire API. In order to have more control, such as returning the header for some origins and not others, you must upgrade to v10 and use the CORS policy feature. For more info, see:

- [Enabling CORS support for an API](#)
- OpenAPI 3.0 editor: [Setting API properties](#)
- OpenAPI 2.0 editor: [Setting API properties](#)

In API Connect 10.0.3.0 or later, the `x-ibm-gateway-cors-allow-credentials-when-no-cors-policy` property works, but only for API definitions which do NOT contain a cors-policy. You cannot define both this property and a CORS Policy on the same API definition.

This property is deprecated in v10: a warning will be written to the log for every CORS request received by an API whose definition contains this property.

For v10, the recommended way to ensure the ACAC header does not appear in CORS responses is to add a CORS policy to the API definition. In v10, any API definition which has a cors-policy defined will NOT return the ACAC header by default. Note that it is possible in the CORS policy rules to explicitly return the ACAC header for some or all origins.

If you move your configuration from Version 2018 to 10.0.3.0 or later, and have this custom property defined in your API definitions, you will not see any change in behavior, other than new warnings in the log. This property will remove the ACAC header for any API definition where it appears.

If, when you move your configuration from Version 2018 to Version 10, you want to have the ACAC header present in the CORS responses, you can leave the API definitions as-is (without the new property defined) and the ACAC header will be returned, as long as a cors-policy is not added to the API definition. Note also that even with a CORS policy defined for an API, the ACAC header can be returned if desired by explicitly allowing it in the rules of the CORS policy.

Note that the `x-ibm-gateway-cors-allow-credentials-when-no-cors-policy` property in an API definition:

- Will cause a commit-time error in v10 if a cors-policy is also present in the same API definition
- Will NOT work in v10 or v2018 to turn ON the ACAC header; it can ONLY be used to turn OFF the header. Explanation:
 - In v2018, ACAC is on by default
 - In v10, without cors-policy, ACAC is on by default
 - In v10, with cors-policy, ACAC can be turned on for specific origins via rules in cors-policy

Upgrading to v10.0.5.3 (or later) from an earlier 10.05.x release.

Additional features related to inter-subsystem communication were added in v10.0.5.3. To enable and configure these features see: [Optional post-upgrade steps for upgrade to 10.0.5.3 \(or later\) from earlier 10.0.5 release.](#)

- **Additional considerations for upgrading analytics from a version prior to 10.0.5.0**
API Connect v10.0.5.0 introduced a new architecture for the Analytics subsystem, so some additional steps are required before and after upgrading from API Connect 10.0.4-ifix3 or 10.0.1.x.

Additional considerations for upgrading analytics from a version prior to 10.0.5.0

API Connect v10.0.5.0 introduced a new architecture for the Analytics subsystem, so some additional steps are required before and after upgrading from API Connect 10.0.4-ifix3 or 10.0.1.x.

The key changes to the Analytics subsystem are highlighted in [What's new in Version 10.0.5.](#)

The most important consideration is that existing analytics data is deleted during the upgrade from 10.0.4-ifix3, and from 10.0.1.7-eus (or higher), and cannot be restored. After upgrading to v10.0.5.0 or later, you will have an empty analytics database. If you want to keep your existing analytics data on an external system, there are three options:

- **Exporting analytics data from the UI**
Before upgrading to v10.0.5, export any analytics data that you wish to keep using the UI.
- **Exporting analytics data by using the REST API**
Before upgrading to v10.0.5, export any analytics data that you want to keep by using the REST API.
- **Offloading analytics data**
An alternative to exporting your analytics data before upgrading to v10.0.5 is to enable analytics offload. However, when enabled this feature only offloads new analytics data. But if your upgrade is not planned for a few days or weeks, you can enable this feature now and avoid the need to export any new analytics data.
- **Remove template overrides from your analytics CR**
If you are upgrading to v10.0.5.4 or previous, remove any analytics CR template overrides. Pre-10.0.5 analytics CR template overrides are not compatible with 10.0.5 and must be removed.

Exporting analytics data from the UI

Before upgrading to v10.0.5, export any analytics data that you wish to keep using the UI.


About this task

Note: This page refers to analytics features that existed in pre-v10.0.5 releases, for the purposes of exporting analytics data before upgrade. Exporting analytics data from v10.0.5 is described here [Exporting analytics from v10.0.5](#)

Use the Analytics Discover tab to search for event data and then export the *hits* (the records in the search results) to a file. Each hit represents one API invocation in the Gateway server. When you view the hits in the UI, the display is limited to 500 records. To view the entire set of hits, export the data to a CSV or JSON Lines file.

Procedure

1. Navigate to the Analytics page and click Discover.
2. Run a search.

Select a predefined search to run, or click  to define your own search, and then run it. Optionally apply filters to refine the results. The export option is only available when a search returns at least one hit.

3. Click Export (this option displays before the search results).
4. In the "Export Hits" panel, select the Filetype for the exported data.
There are two options for the file's type:

- a. JSON Lines (.txt): Exports data to a .txt file that contains a complete JSON record for each hit.
- b. Comma-separated Values (.csv): Exports to a .csv file that can be viewed either in a text editor, a spreadsheet, or some other CSV viewer.
If you intend to view the .csv file in a viewer that does not support special characters, select the following options as needed to ensure compatibility with your viewer:
 - i. Escape double quotes: Insert an additional quotation mark (") as an escape character before each " that is already included in the file.
 - ii. Escape special characters: Insert a quotation mark (") as an escape character before each of the following characters: = _ + % @

The content of the API event records in the CSV file depends on the logging policy that is set for the operation. If the logging policy includes the HTTP headers and the payload, then these details are included in the API event record. Where the payload is large, it might take longer to complete the export of the analytics data. For more information, see [activity-log.policy](#). For information about the fields that are included in the CSV file, see [API event records](#).

5. In the panel, click Export to start the export operation.
6. When the "The export has completed successfully." message displays, click Download to save the file, or click Another one? to run a new export operation. (Remember to save the current export results before running a new export).

Attention:

- If your browser is configured to block pop-ups, the download might be blocked. The precise behavior varies across browsers, but if you see a notification about "blocked pop-ups", reconfigure the browser to always allow pop-ups for the API Manager host address. Then check the "Export Hits" panel for the Download option, which completes the operation. If you do not see a Download option, repeat the export operation.
- If you navigate away from the Analytics interface before the exported file is downloaded, the download operation is suspended. If you return to Analytics before any other export operation begins and before the file expires, you might see a Download option that allows you to finish downloading the file. Otherwise, you must repeat the entire export operation.

- Each downloaded file is saved to the download location that is configured for your browser, using a randomly generated file name. If you intend to export more than once, and you are exporting to the same location each time, you might want to rename each file with a meaningful name.
- [Exporting aggregated analytics data from a visualization](#)
From a visualization, you can view the raw aggregated data from which the graphical representation is constructed, and then export that data to a CSV file.

Exporting aggregated analytics data from a visualization

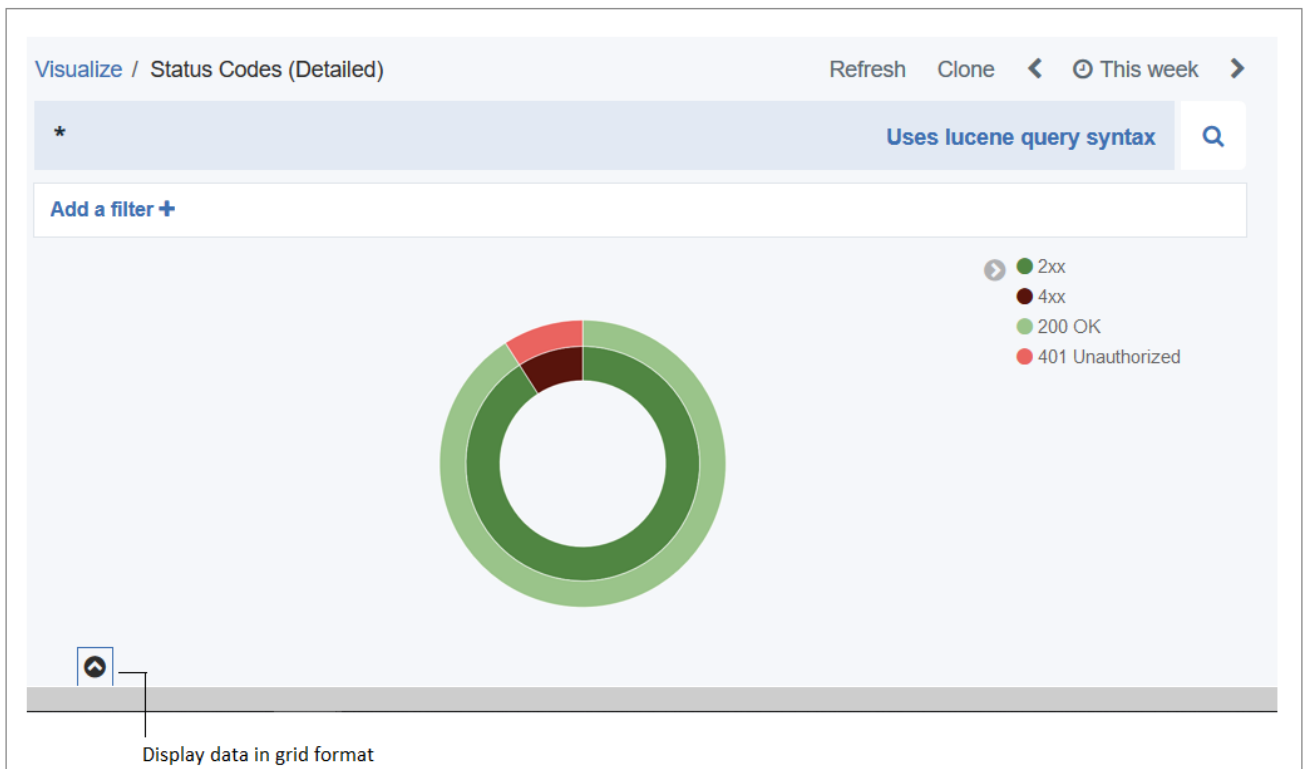
From a visualization, you can view the raw aggregated data from which the graphical representation is constructed, and then export that data to a CSV file.


Procedure

Note: This page refers to analytics features that existed in pre-v10.0.5 releases, for the purposes of exporting analytics data before upgrade. Exporting analytics data from v10.0.5 is described here [Exporting analytics from v10.0.5](#)

To export analytics data from a visualization, complete the following steps.

1. Navigate to the Analytics page and click Visualize.
2. Select a visualization by clicking it.
For example, you might select the visualization called "Status Codes (Detailed)" as shown in the following image:



3. Display the data in grid format by clicking the  that displays after the visualization.
In the resulting grid, a row is displayed for each aggregated data set that matches the search criteria configured in the visualization's aggregation builder. The raw data shown in the columns and rows depends on the type of visualization accessed and the aggregations configured for that visualization.

Visualize / Status Codes (Detailed) Refresh Clone < This week >

* Uses lucene query syntax Q

Add a filter +

Table

filters	Count	status_code.keyword: Descending	Count
1xx	0		
2xx	20	200 OK	20
3xx	0		
4xx	2	401 Unauthorized	2
5xx	0		

Export: Raw Formatted

Page Size 10

Display in graph format Export options

4. Export a tabular format of the raw data that is displayed in the grid:
5. When prompted, choose to save the file, which is named *visualization_name.csv* by default.
The file is saved to the download location that is configured for your browser. If you intend to export more than once from the same or another visualization, and you are exporting to the same location each time, you might want to rename each file with a meaningful name.

Exporting analytics data by using the REST API

Before upgrading to v10.0.5, export any analytics data that you want to keep by using the REST API.

Before you begin

API Connect REST API calls are authenticated by using a bearer token in the authorization header. Use the toolkit credentials to request a bearer token. Toolkit credentials can be found in the API Manager UI:

1. Log in to the API Manager UI.
2. From the home page, click the Download toolkit tile.
3. Download the Toolkit credentials.
4. Open the downloaded `credentials.json` file, example:

```
...
"toolkit": {
  "endpoint": "https://mgmt.api.example.com/api",
  "client_id": "7409693f-f726-48b4-8909-7c0d26f13e81",
  "client_secret": "5feeb0be-17f8-41a4-96d6-d40f33d69ef6"
},
...
```

Take note of the `toolkit.client_id`, `toolkit.client_secret`, and `toolkit.endpoint` from this file. You will use these to get your bearer token.

About this task

The Analytics REST API is documented in the [API Connect v10.0.5LATEST](#) API Explorer. You can export the data catalog-by-catalog using `/analytics/catalogs/{org}/{catalog}/analytics-services/{analytics-service}/events`

Procedure

1. Use your provider organization credentials and `client_id/secret` to request a bearer token:

```
curl -v -k -X POST -d '{"username": "<username>", "password": "<password>", "realm": "<realm>", "client_id": "<client_id>", "client_secret": "<client_secret>", "grant_type": "password"}' -H 'Content-Type: application/json' -H
```



```
'Accept: application/json' https://<management server platform api endpoint>/token
```

Where:

- <username> is the provider org owner.
- <password> is the password for the specified <username>.
- <realm> is the user registry realm for the specified <username>. The default provider organization user registry realm is `provider/default-idp-2`.
- <client_id> is the client ID from the "toolkit" section of the `credentials.json` file, or as provided by your API Connect cloud administrator.
- <client_secret> is the client secret from the "toolkit" section of the `credentials.json` file, or as provided by your API Connect cloud administrator.
- <management server platform api endpoint> is the platform API endpoint from the "toolkit" section of the `credentials.json` file, or as provided by your API Connect cloud administrator.

The bearer token is returned in the `access_token` property:

```
{
  "access_token": "<bearer token>",
  "token_type": "Bearer",
  "expires_in": 28800
}
```

Note the `expires_in` value, which is the number of seconds before the token expires. After expiry you need to request a new bearer token.

2. Use the returned bearer token to call the analytics REST API:

```
curl -v -k -H 'Accept: application/json' -H 'Authorization: Bearer <bearer token>' -X GET --url 'https://<management server platform api endpoint>/analytics/catalogs/<provider org name>/<catalog name>/analytics-services/<analytics service name>/events?after=now-7d'
```

Where:

- <management server platform api endpoint> is the `toolkit.endpoint`, but with the `/api` at the end replaced with `/analytics` to access the analytics APIs.

In this example, the REST API call returns the last 7 days (note the `?after=now-7d` at end of the URL) of event data for the specified catalog:

```
{ "total": 2, "found": true, "events": [ { "id": "AYGp1P_4umraqLKQkId3", "event": { "immediate_client_ip": "9.145.95.209", "developer_org_name": "", "resource_id": ":GET:/hola-mundo", "developer_org_id": "", "log_policy": "activity" ...
```

Note: For large data sets it might be necessary to make the REST call repeatedly, specifying a different time range for each call. For example, to capture events logged during a specific 30-minute interval, use: `?after=2022-06-28T09:00:00&before=2022-06-28T09:30:00`

3. Additional filters can be used as described in the REST API documentation for the call to `GET /analytics/catalogs/{org}/{catalog}/analytics-services/{analytics-service}/events`

Offloading analytics data

An alternative to exporting your analytics data before upgrading to v10.0.5 is to enable analytics offload. However, when enabled this feature only offloads new analytics data. But if your upgrade is not planned for a few days or weeks, you can enable this feature now and avoid the need to export any new analytics data.

The steps for configuring analytics offload are covered in the v10.0.4 documentation: <https://www.ibm.com/docs/en/api-connect/10.0.x?topic=topology-offloading-data-third-party-system>

Remove template overrides from your analytics CR

If you are upgrading to v10.0.5.4 or previous, remove any analytics CR template overrides. Pre-10.0.5 analytics CR template overrides are not compatible with 10.0.5 and must be removed.

If you are upgrading to v10.0.5.4 or a previous v10.0.5 fix pack, then review edit your analytics CR and if it has a `template` section, remove this section. For example:

```
kubectl edit a7s -n <namespace>
```

Remove the `template` section:

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsCluster
...
spec:
...
  template: # REMOVE THIS SECTION
  - name: storage-data
    containers:
    - name: storage-data
      env:
      - name: ES_JAVA_OPTS
        value: "-Xms2g -Xmx2g"
      resources:
      requests:
        memory: 3Gi
      limits:
        memory: 4Gi
```

In this example, the template section specified custom memory allocations for internal analytics processes. These settings are applicable only to pre-v10.0.5 analytics deployments. v10.0.5 has additional deployment profiles that provide increased memory allocations: [Analytics component deployment profile limits](#).

Upgrading subsystems on native Kubernetes

Upgrade your API Connect subsystems to the latest version on native Kubernetes.

Before you begin

- To ensure operator compatibility, upgrade API Connect management subsystem first and DataPower gateway subsystem second. This requirement applies to all upgrade scenarios.
The Gateway subsystem remains available during the upgrade of the Management, Portal, and Analytics subsystems.
- Verify that your installed Kubernetes version is supported by the API Connect version that you are upgrading to. See [IBM API Connect Version 10 software product compatibility requirements](#). If your Kubernetes version is older than the minimum supported version for your target API Connect version, then upgrade Kubernetes first.
Note: If you are upgrading from an old API Connect version, it is possible that it does not support the minimum supported Kubernetes version that is required by the latest API Connect release. If so, one or more intermediate upgrades of API Connect might be required to ensure that API Connect is always running on a supported Kubernetes version.

Procedure

- Complete the prerequisites:
 - Ensure you are upgrading from a supported version and reviewed the upgrade requirements and limitations. See [Upgrade considerations on native Kubernetes](#).
 - Verify that the `pgcluster` is healthy:
 - Get the name of the `pgcluster`:

```
kubectl get pgcluster -n <APIC_namespace>
```

The response displays the name of the postgres cluster running in the specified namespace.
 - Check the status of the `pgcluster`:

```
kubectl get pgcluster <pgcluster_name> -n <APIC_namespace> -o yaml | grep status -A 2 | tail -n3
```

The response for a healthy `pgcluster` looks like the following example, where the state is `Initialized`:

```
status:
  message: Cluster has been initialized
  state: pgcluster Initialized
```

Important: If the `pgcluster` returns any other state, it is not healthy and an upgrade will fail.
 - If there are any on-going backup or restore jobs, wait until they complete and then check the status again. Do not proceed with the upgrade until the status is `Initialized`.
 - If all of the background jobs complete but the `pgcluster` remains unhealthy, contact IBM Support for assistance.
 - Backup the current deployment. Wait until the backup completes before starting the upgrade.
 - Do not start an upgrade if a backup is scheduled to run within a few hours.
 - Do not perform maintenance tasks such as rotating key-certificates, restoring from a backup, or starting a new backup, at any time while the upgrade process is running.
 - If you used any microservice image overrides in the management CR during a fresh install, you must remove the image overrides prior to upgrade.
Important: When upgrading, if you used any microservice image overrides in the management CR during a fresh install then prior to upgrade these image overrides will be automatically removed by the operator during upgrade. You can apply them again after the upgrade is complete.
 - Back up API test data in the Automated API behavior testing application as explained in [Backing up API test data](#). After the upgrade is complete, you can migrate the data as explained in [Restoring API test data](#).
- Run the pre-upgrade health check:
 - Verify that the `apicops` utility is installed by running the following command to check the current version of the utility:

```
apicops --version
```

If the response indicates that `apicops` is not available, install it now. See [The API Connect operations tool: apicops](#) in the API Connect documentation.
 - Run the following command to set the KUBECONFIG environment.

```
export KUBECONFIG=</path/to/kubeconfig>
```
 - Run the following command to execute the pre-upgrade script:

```
apicops version:pre-upgrade -n <namespace>
```

If the system is healthy, the results will not include any errors.

Note: This command asks for the Cloud Manager admin password in order to make a call to the platform API. If you do not have this password, or the platform API URL is not accessible from your `apicops` location, then this part of the pre-upgrade check can be skipped by adding the argument `--no-topology`. The topology check output that is produced by this API call is typically only required for debugging purposes, when the `apicops version:pre-upgrade` returns output that suggests the system is unhealthy.
- Obtain the API Connect files from IBM Fix Central.
From the [IBM Fix Central](#) site, download the Docker image-tool file of the API Connect subsystems. Next, you will upload the image-tool file to your Docker local registry. If necessary, you can populate a remote container registry with repositories. Then you can push the images from the local registry to the remote registry.

You will also download the Kubernetes operators, API Connect Custom Resource (CR) templates, and Certificate Manager, for use during deployment configuration.

The following files are used for deployment on native Kubernetes:

IBM® API Connect <version> for Containers

Docker images for all API Connect subsystems

IBM® API Connect <version> Operator Release Files for Containers

Kubernetes operators and API Connect Custom Resource (CR) templates

IBM® API Connect <version> Toolkit for <operating_system_type>

Toolkit command line utility. Packaged standalone, or with API Designer or Loopback:

- IBM® API Connect <version> Toolkit for <operating_system_type>
- IBM® API Connect <version> Toolkit with Loopback for <operating_system_type>
- IBM® API Connect <version> Toolkit Designer with Loopback for <operating_system_type>

Not required during initial installation. After installation, you can download directly from the Cloud Manager UI and API Manager UI. See [Installing the toolkit](#).

IBM® API Connect <version> Local Test Environment

Optional test environment. See [Testing an API with the Local Test Environment](#)

IBM® API Connect <version> Security Signature Bundle File

Checksum files that you can use to verify the integrity of your downloads.

4. Next, upload the image files that you obtained from Fix Central in Step 3.
 - a. Load the image-tool image for the new version into your Docker local registry:

```
docker load < apiconnect-image-tool-<version>.tar.gz
```

Ensure that the registry has sufficient disk space for the files.

- b. If your Docker registry requires repositories to be created before images can be pushed, create the repositories for each of the images listed by the image tool. If your Docker registry does not require creation of repositories, skip this step and go to Step 4.c.
 - i. Run the following command to get a list of the images from image-tool:

```
docker run --rm apiconnect-image-tool-<version> version --images
```

- ii. From the output of each entry of the form <image-name>:<image-tag>, use your Docker registry repository creation command to create a repository for <image-name>.

For example in the case of AWS ECR the command would be for each <image-name>:

```
aws ecr create-repository --repository-name <image-name>
```

- c. Upload the image:

- If you do not need to authenticate with the docker registry, use:

```
docker run --rm apiconnect-image-tool-<version> upload <registry-url>
```

- Otherwise, if your docker registry accepts authentication with username and password arguments, use:

```
docker run --rm apiconnect-image-tool-<version> upload <registry-url> --username <username> --password <password>
```

- Otherwise, such as with IBM Container Registry, if you need the image-tool to use your local Docker credentials, first authenticate with your Docker registry, then upload images with the command:

```
docker run --rm -v ~/.docker:/root/.docker --user 0 apiconnect-image-tool-<version> upload <registry-url>
```

Review the following installation notes as appropriate for your environment:

- [Alternative Docker command for macOS if Docker is configured to use osxkeychain](#)
- [Docker authentication notes](#)
- [Providing a certificate for verification, or disabling TLS verification](#)

5. If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher) and you want to retain your analytics data, then you must export it before running the upgrade.

For instructions on exporting data, see [Additional considerations for upgrading analytics from a version prior to 10.0.5.0](#).

6. If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher): Disassociate and delete your Analytics services.

- a. In Cloud Manager, click Topology.
- b. In the section for the Availability Zone that contains the Analytics service, locate the Gateway service that the Analytics service is associated with.
- c. Click the actions menu, and select Unassociate analytics service.
Remember to disassociate each Analytics service from all Gateways.
- d. In the section for the Availability Zone that contains the Analytics services, locate each Analytics service and click Delete.

7. Download and decompress IBM API Connect <version> Operator Release Files for Containers.

Make a directory called **helper_files** and extract the contents of **helper_files.zip** from the **release_files.zip** into **helper_files**.

8. Apply the new CRDs from the version you just extracted:

```
kubectl apply -f ibm-apiconnect-crds.yaml
```

9. Apply the new DataPower Operator YAML into the namespace where the DataPower Operator is running.

- a. Verify that the correct namespace is referenced in the **ibm-datapower.yaml** file:

If your product was not deployed in the namespace called **default**, open the **ibm-datapower.yaml** file in a text editor and replace all instances of **default** with the appropriate name of the namespace where you deployed the product.

- b. Specify the location of the **datapower-operator** image:

Open the **ibm-datapower.yaml** file in a text editor and locate the **image:** key in the **containers** section of the deployment file (immediately after **imagePullSecrets:**). Replace the value of the **image:** key with the location of the **datapower-operator** image, either uploaded to your own registry or pulled from a public registry.

- c. Run the following command:

```
kubectl apply -f ibm-datapower.yaml -n <namespace>
```

The Gateway CR goes to **Pending** state when the operator is updated, and then changes to **Running** after installation of the API Connect operator in the next step.

Note: There is a known issue on Kubernetes version 1.19.4 or higher that can cause the DataPower operator to fail to start. In this case, the DataPower Operator pods can fail to schedule, and will display the status message: **no nodes match pod topology spread constraints (missing required label)**. For example:

```
0/15 nodes are available: 12 node(s) didn't match pod topology spread constraints (missing required label), 3 node(s) had taint {node-role.kubernetes.io/master: }, that the pod didn't tolerate.
```

You can work around the issue by editing the DataPower operator deployment and re-applying it, as follows:

- i. Delete the DataPower operator deployment, if deployed already:

```
kubectl delete -f ibm-datapower.yaml -n <namespace>
```

- ii. Open `ibm-datapower.yaml`, and locate the `topologySpreadConstraints`: section. For example:

```
topologySpreadConstraints:
- maxSkew: 1
  topologyKey: zone
  whenUnsatisfiable: DoNotSchedule
```

- iii. Replace the values for `topologyKey`: and `whenUnsatisfiable`: with the corrected values shown in the example below:

```
topologySpreadConstraints:
- maxSkew: 1
  topologyKey: topology.kubernetes.io/zone
  whenUnsatisfiable: ScheduleAnyway
```

- iv. Save `ibm-datapower.yaml` and deploy the file to the cluster:

```
kubectl apply -f ibm-datapower.yaml -n <namespace>
```

10. Upgrade cert-manager to version 1.9.1.

- a. Run the following command to back up the existing certificates and issuers to a file called `backup.yaml`:

```
kubectl get --all-namespaces -oyaml issuer,clusterissuer,cert,secret > backup.yaml
```

- b. Run the following command to determine the version of the existing cert-manager:

```
kubectl get crds certificates.cert-manager.io -o jsonpath='{.metadata.labels.app\.kubernetes\.io\/version}'
```

- c. If the current version is 1.5.1 or 1.7.1, run the following command to upgrade cert-manager to version 1.9.1:

```
kubectl apply -f helper-files/cert-manager-1.9.1.yaml
```

11. If you used customized internal certificates, and are upgrading from 10.0.5.1: The `helper_files/custom-certs-internal.yaml` includes a new certificate called `dbClientPrimaryuser`. This new certificate must be created before you update the Management CR.

Follow the instructions in [Generate custom internal certificates](#) to update the `helper_files/custom-certs-internal.yaml` file to match your namespace and site name, and then apply the new `yaml` file with the following command:

```
kubectl apply -f custom-certs-internal.yaml -n <namespace>
```

Alternatively, add the new certificate to your existing `custom-certs-internal.yaml` and apply it, updating the namespace to match your deployment:

```
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: db-client-primaryuser
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "db-client-primaryuser"
  }
spec:
  commonName: primaryuser
  secretName: db-client-primaryuser
  dnsNames:
  - "*.<namespace>"
  - "*.<namespace>.svc"
  - "primaryuser.<namespace>.svc"
  - "primaryuser"
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
---
```

12. Clean up the webhook configuration before deploying the newer API Connect operator:

- a. Run the following command to get the webhook configuration:

```
kubectl get mutatingwebhookconfiguration,validatingwebhookconfiguration | grep ibm-apiconnect
```

- b. Use the `kubectl delete` command to delete the webhook configuration; for example:

```
kubectl delete mutatingwebhookconfiguration/ibm-apiconnect-<namespace>-mutating-webhook-configuration
kubectl delete validatingwebhookconfiguration/ibm-apiconnect-<namespace>-validating-webhook-configuration
```

13. Apply the new API Connect operator YAML into the namespace where the API Connect operator is running.

- For single namespace deployment:
 - If the operator is not running in the `default` namespace, open the `ibm-apiconnect.yaml` file in a text editor, and then replace all references to `default` with the name of the namespace where you deployed API Connect.
Note: Skip this step if you are using Operator Lifecycle Manager (OLM).
 - Open `ibm-apiconnect.yaml` in a text editor. Replace the value of each `image:` key with the location of the `apiconnect` operator images (from the `ibm-apiconnect` container and the `ibm-apiconnect-init` container), either uploaded to your own registry or pulled from a public registry.
 - Run the following command:

```
kubectl apply -f ibm-apiconnect.yaml -n <namespace>
```

- For multi-namespace deployment:
 - Locate and open the newly downloaded `ibm-apiconnect-distributed.yaml` in a text editor of choice. Then, find and replace each occurrence of `$OPERATOR_NAMESPACE` with the desired namespace for the deployment.
 - Also in `ibm-apiconnect-distributed.yaml`, locate the `image:` keys in the containers sections of the deployment yaml right below `imagePullSecrets:`. Replace the placeholder values `REPLACE-DOCKER-REGISTRY` of the `image:` keys with the docker registry host location of the API Connect operator image (either uploaded to own registry or pulled from public registry).
 - Install `ibm-apiconnect-distributed.yaml` with the following command

```
kubectl apply -f ibm-apiconnect-distributed.yaml
```

14. Verify that the `ibm-datapower-operator` and the `ibm-apiconnect` operators are restarted.

15. Ensure that the `apiconnect` operator recreated the necessary microservices:

Be sure to complete this step before attempting to upgrade the operands.

```
kubectl get apic -n <namespace>
```

16. Upgrade the operands (subsystems):

- [Upgrading Management subsystem on native Kubernetes](#)
- [Upgrading Portal subsystem on native Kubernetes](#)
- [Upgrading Analytics subsystem on native Kubernetes](#)
- [Upgrading Gateway subsystem on native Kubernetes](#)

17. Verify that the upgraded subsystems report as `Running` and the `RECONCILED VERSION` displays the new version of API Connect.

Run the following command:

```
kubectl get apic --all-namespaces
```

Example response:

NAME	READY	STATUS	VERSION	RECONCILED VERSION	AGE
analyticscluster.analytics.apiconnect.ibm.com/analytics		8/8	Running	10.0.5.4	10.0.5.4-1074 121m
NAME	PHASE	READY	SUMMARY	VERSION	AGE
datapowerservice.datapower.ibm.com/gwl	Running	True	StatefulSet replicas ready: 1/1	10.0.5.4	100m
NAME	PHASE	LAST EVENT	WORK PENDING	WORK IN-PROGRESS	AGE
datapowermonitor.datapower.ibm.com/gwl	Running		false	false	100m
NAME	READY	STATUS	VERSION	RECONCILED VERSION	AGE
gatewaycluster.gateway.apiconnect.ibm.com/gwl	2/2	Running	10.0.5.4	10.0.5.4-1074	100m
NAME	READY	STATUS	VERSION	RECONCILED VERSION	AGE
managementcluster.management.apiconnect.ibm.com/ml	16/16	Running	10.0.5.4	110.0.5.4-1074	162m
NAME	READY	STATUS	VERSION	RECONCILED VERSION	AGE
portalcluster.portal.apiconnect.ibm.com/portal	3/3	Running	10.0.5.4	10.0.5.4-1074	139m

Important: If you need to restart the deployment, wait until all Portal sites complete the upgrade. Run the following commands to check the status of the sites:

a. Log in as an admin user:

```
apic login -s <server_name> --realm admin/default-idp-1 --username admin --password <password>
```

b. Get the portal service ID and endpoint:

```
apic portal-services:get -o admin -s <management_server_endpoint> \
  --availability-zone availability-zone-default <portal-service-name> \
  --output - --format json
```

c. List the sites:

```
apic --mode portaladmin sites:list -s <management_server_endpoint> \
  --portal_service_name <portal-service-name> \
  --format json
```

Any sites currently upgrading display the `UPGRADING` status; any site that completed its upgrade displays the `INSTALLED` status and the new platform version. Verify that all sites display the `INSTALLED` status before proceeding.

For more information on the `sites` command, see [apic sites:list](#) and [Using the sites commands](#).

d. After all sites are in `INSTALLED` state and have the new platform listed, run:

```
apic --mode portaladmin platforms:list -s <server_name> --portal_service_name <portal_service_name>
```

Verify that the new version of the platform is the only platform listed.

For more information on the `platforms` command, see [apic platforms:list](#) and [Using the platforms commands](#).

18. If you are upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher): Enable analytics as explained in [Enabling Analytics after upgrading](#).
19. If you are upgrading from v10.0.5.3 or earlier: Update the certificates with the ingress-issuer-v1.yaml from your target release.

```
kubectl apply -f helper_files/ingress-issuer-v1.yaml -n <namespace>
```

For a multi-namespace deployment, repeat the command for each namespace.

20. Optional: For the optional components API Connect Toolkit and API Connect Local Test Environment, install the latest version of each after you complete the upgrade of the subsystems.
21. Optional: Configure additional features related to inter-subsystem communication security, such as CA verification and JWT security: [Optional post-upgrade steps for upgrade to 10.0.5.3 \(or later\) from earlier 10.0.5 release](#).

- [Upgrading Management subsystem on native Kubernetes](#)

Upgrade the management subsystem to the latest version of API Connect.

- [Upgrading Portal subsystem on native Kubernetes](#)

Upgrade the Portal subsystem to the latest version of API Connect.

- [Upgrading Analytics subsystem on native Kubernetes](#)

Upgrade the Analytics subsystem to the latest version of API Connect.

- [Upgrading Gateway subsystem on native Kubernetes](#)

Upgrade the Gateway subsystem to the latest version of API Connect.

- [Enabling gateway peering and verifying cluster status](#)

To complete the operand upgrade, restart the gateway pods to enable gateway peering.

- [Enabling Analytics after upgrading](#)

After the upgrade successfully completes, re-enable Analytics data collection.

- [Optional post-upgrade steps for upgrade to 10.0.5.3 \(or later\) from earlier 10.0.5 release](#)

Version 10.0.5.3 introduces some new features that are related to inter-subsystem communication. This topic explains how to configure these features when you upgrade from a previous 10.0.5 release to v10.0.5.3 or later.

Upgrading Management subsystem on native Kubernetes

Upgrade the management subsystem to the latest version of API Connect.

Before you begin

Complete all steps in [Upgrading subsystems on native Kubernetes](#) prior to the step that links to this topic.

About this task

When upgrading to a new mod release the version must be changed to the latest mod release version on each CR. This change will be picked up by the operator, and the operator will then start the upgrade.

Procedure

1. Update the management CR for the new version of API Connect.

- a. Run the following command to edit the CR:

```
kubectl -n $NAMESPACE edit mgmt <CLUSTERNAME>
```

where *CLUSTERNAME* is the name specified in the subsystem CR at installation time.

- b. **If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher):** In the `spec` section, add a new `allowUpgrade` attribute and set it to `true`:

```
spec:
  allowUpgrade: true
```

The `allowUpgrade` attribute enables the upgrade to 10.0.5.0 or later. Because the upgrade to 10.0.5.x deletes your existing analytics data, the attribute is required to prevent an accidental upgrade. See [Additional considerations for upgrading analytics from a version prior to 10.0.5.0](#). After the Management subsystem has been upgraded, the Analytics subsystem must also be upgraded.

- c. Update the API Connect `version` in the CR:

For example:

```
version: 10.0.5.4
```

- d. If you are upgrading to a version of API Connect that requires a new license, update the license value now.

For example:

```
license: L-GVEN-GFUPVE
```

For the list of licenses, see [API Connect licenses](#).

- e. Run the following command to save and apply the CR: `wq`
When you save the updated CR, the upgrade starts automatically.

Note: If you see an error message when you attempt to save the CR, check if it is one of the following known issues:

- Webhook error for incorrect license.

If you did not update the license ID in the CR, then when you save your changes, the following webhook error might display:

```
admission webhook "vmanagementcluster.kb.io" denied the request:
ManagementCluster.management.apiconnect.ibm.com "management" is invalid:
spec.license.license: Invalid value: "L-RJON-BYGHM4":
```

License L-RJON-BYGHM4 is invalid for the chosen version 10.0.5.4.
Please refer license document <https://ibm.biz/apiclicenses>

To resolve the error, see [API Connect licenses](#) for the list of the available license IDs and select the appropriate License IDs for your deployment. Update the CR with the new license value as in the following example, and then save and apply your changes again.

- Webhook error: Original postgres primary is running as replica
If you see the error message:

Original PostgreSQL primary is running as replica,
please perform failover to original primary before attempting upgrade. Refer - <https://ibm.biz/BdPLWU>"

you must complete a Postgres failover before the upgrade can proceed, see [Postgres failover steps](#). After you apply the Postgres failover steps, the upgrade resumes automatically.

- Webhook error: Original PostgreSQL primary not found
If you see the error message:

Original PostgreSQL primary not found. Upgrade is blocked.
Set `apiconnect-operator/db-primary-not-found-allow-upgrade=true` annotation to unblock upgrade.
Perform db backup and restore after upgrade to restore original primary.

take the following actions to complete the upgrade and fix the cause of the error message:

- Edit the management CR:

```
kubectl -n $NAMESPACE edit mgmt <CLUSTERNAME>
```

- Add the following annotation:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementCluster
metadata:
  annotations:
    apiconnect-operator/db-primary-not-found-allow-upgrade=true
    ...
```

- Continue with the upgrade. When the upgrade is complete, the management CR reports the warning:

Original PostgreSQL primary not found, perform db backup and restore to restore original primary.

- Take a new [management database backup](#).
- Immediately [restore](#) from the new backup taken in the previous step. The action of taking and restoring a management backup results in the establishment of a new Postgres primary, eliminating the CR warning message. Be careful to restore from the backup that is taken after the upgrade, and not from a backup taken before upgrade.

2. Run the following command to verify that the upgrade was successful:

```
kubectl get apic -n <namespace>
```

Example output after upgrading the management subsystem:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
managementcluster.management.apiconnect.ibm.com/ml	16/16	Running	<version>	<version-build>	162m	
NAME	AGE	STATUS	MESSAGE			
managementdbupgrade.management.apiconnect.ibm.com/management-up-8c28g	13m	Running	Upgrade is running (DB Schema/data are up-to-date)			

3. Restart all nats-server pods by running the following command:

```
kubectl -n <namespace> delete po -l app.kubernetes.io/name=natscluster
```

What to do next

Upgrade your Portal subsystem: [Upgrading Portal subsystem on native Kubernetes](#).

If you are upgrading from v10.0.5.1 or earlier to v10.0.5.2: The change in deployment profile CPU and memory limits that are introduced in 10.0.5.2 (see [New deployment profiles and CPU licensing](#)) can result in a change in the performance of your Management component. If you notice any obvious reduction in performance of the Management UI or toolkit CLI where you have multiple concurrent users, open a support case.

Upgrading Portal subsystem on native Kubernetes

Upgrade the Portal subsystem to the latest version of API Connect.

Before you begin

- Complete all steps in [Upgrading subsystems on native Kubernetes](#) prior to the step that links to this topic.
- If you are upgrading from 10.0.5.2 or earlier: If you did not verify that your Portal customizations are compatible with Drupal 10, do that now. In API Connect 10.0.5.3, the Developer Portal moved from Drupal 9 to Drupal 10 (this upgrade also requires PHP 8.1). The upgrade tooling will update your Developer Portal sites; however, if you have any custom modules or themes, it is your responsibility to ensure their compatibility with Drupal 10 and PHP 8.1 before starting the upgrade. Review the [Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10](#) to ensure that any customizations to the Developer Portal are compatible with Drupal 10 and PHP 8.1.

About this task

When upgrading to a new mod release, the version must be changed to the latest mod release version on each CR. This change will be picked up by the operator, and the operator will then start the upgrade.

Procedure

1. Update the portal CR for the new version of API Connect.

- a. Run the following command to edit the CR:

```
kubectl -n $NAMESPACE edit ptl <CLUSTERNAME>
```

where `CLUSTERNAME` is the name specified in the subsystem CR at installation time.

- b. Update the API Connect `version` in the CR:

For example:

```
version: 10.0.5.4
```

- c. If you are upgrading to a version of API Connect that requires a new license, update the license value now.

For example:

```
license: L-GVEN-GFUPVE
```

For the list of licenses, see [API Connect licenses](#).

- d. Run the following command to save and apply the CR: `wq`

When you save the updated CR, the upgrade starts automatically.

Known issue: Webhook error for incorrect license.

If you did not update the license ID in the CR, then when you save your changes, the following webhook error might display:

```
admission webhook "vmanagementcluster.kb.io" denied the request:
ManagementCluster.management.apiconnect.ibm.com "management" is invalid:
spec.license.license: Invalid value: "L-RJON-BYGHM4":
License L-RJON-BYGHM4 is invalid for the chosen version 10.0.5.4.
Please refer license document https://ibm.biz/apiclicenses
```

To resolve the error, see [API Connect licenses](#) for the list of the available license IDs and select the appropriate License IDs for your deployment. Update the CR with the new license value as in the following example, and then save and apply your changes again.

2. Run the following command to verify that the upgrade was successful:

```
kubectl get apic -n <namespace>
```

Example output after upgrading the portal subsystem:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	MESSAGE
portalcluster.portal.apiconnect.ibm.com/portal	3/3	Running	<version>	<version>		Site Upgrades
Executing: (1) Queued(5) Failed(0) Success(0) 22h						

Note: After the portal subsystem code upgrade is complete, each portal site is upgraded. You can monitor the site upgrade progress from the `MESSAGE` column in the "kubectl get apic" output. You can still use the portal while sites are upgrading, although a maintenance page is shown for any sites that are being upgraded. When the site upgrades are complete the "kubectl get apic" output will show how many sites the portal is serving:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	MESSAGE	AGE
portal	3/3	Running	<version>	<version>		Serving 2 sites	22h

3. Optional: **If upgrading to v10.0.5.3 (or later) from an earlier 10.0.5.x release:** Review and configure the new inter-subsystem communication features: [Optional post-upgrade steps for upgrade to 10.0.5.3 \(or later\) from earlier 10.0.5 release](#).

Upgrading Analytics subsystem on native Kubernetes

Upgrade the Analytics subsystem to the latest version of API Connect.

About this task

Attention: Upgrading to v10.0.5.x from v10.0.4-ifix3 or later, or v10.0.1.7 or later, will result in the deletion of existing analytics data. If you want to retain your analytics data then you must export it before upgrade. For instructions on exporting see [Additional considerations for upgrading analytics from a version prior to 10.0.5.0](#). When upgrading to a new mod release, the version must be changed to the latest mod release version on each CR. This change will be picked up by the operator, and the operator will then start the upgrade.

Procedure

1. Update the analytics CR for the new version of API Connect.

- a. Run the following command to edit the CR:

```
kubectl -n $NAMESPACE edit a7s <CLUSTERNAME>
```

where `CLUSTERNAME` is the name specified in the subsystem CR at installation time.

- b. **If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher):** Add an `allowUpgrade` attribute and set it to `true`:


```
spec:
  allowUpgrade: true
```

The `allowUpgrade` attribute enables the upgrade to 10.0.5.x. Because the upgrade to 10.0.5.x deletes your existing analytics data, the attribute is required to prevent an accidental upgrade. You must also delete any `template` section from your analytics CR. See [Additional considerations for upgrading analytics from a version prior to 10.0.5.0](#)

- c. Update the API Connect `version` in the CR:
For example:

```
version: 10.0.5.4
```

- d. If you are upgrading to a version of API Connect that requires a new license, update the license value now.
For example:

```
license: L-RJON-CEBLEH
```

For the list of licenses, see [API Connect licenses](#).

- e. Save your changes to the CR.
When you save the updated CR, the upgrade starts automatically.

Known issue: Webhook error for incorrect license.

If you did not update the license ID in the CR, then when you save your changes, the following webhook error might display:

```
admission webhook "vmanagementcluster.kb.io" denied the request:
ManagementCluster.management.apiconnect.ibm.com "management" is invalid:
spec.license.license: Invalid value: "L-RJON-BYGHM4":
License L-RJON-BYGHM4 is invalid for the chosen version 10.0.5.4.
Please refer license document https://ibm.biz/apiclicenses
```

To resolve the error, see [API Connect licenses](#) for the list of the available license IDs and select the appropriate License IDs for your deployment. Update the CR with the new license value as in the following example, and then save and apply your changes again.

2. Run the following command to verify that the upgrade was successful:

```
kubectl get apic -n <namespace>
```

Example output after upgrading the analytics subsystem:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION
AGE					
analyticscluster.analytics.apiconnect.ibm.com/analytics	5/5	Running	<version>	<version-build>	121m

3. Optional: **If upgrading to v10.0.5.3 (or later) from an earlier 10.0.5.x release:** Review and configure the new inter-subsystem communication features: [Optional post-upgrade steps for upgrade to 10.0.5.3 \(or later\) from earlier 10.0.5 release](#).
4. **If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher):** Re-enable analytics data collection as explained in [Enabling Analytics after upgrading](#).
If you are upgrading from 10.0.5.x, skip this step.

Upgrading Gateway subsystem on native Kubernetes

Upgrade the Gateway subsystem to the latest version of API Connect.

Before you begin

- Complete all steps in [Upgrading subsystems on native Kubernetes](#) prior to the step that links to this topic.
- Ensure your upgrade path is supported.
- To ensure operator compatibility, upgrade the API Connect management subsystem before you upgrade the DataPower gateway subsystem. This requirement applies to all upgrade scenarios.
- When you upgrade a cluster of gateway pods in Kubernetes, a small number of API transactions may fail. During the upgrade, Kubernetes removes the pod from the load balancer configuration, deletes the pod and then starts a new pod. The steps are repeated for each pod. Socket hang ups occur on transactions that are in process at the time the pod is killed.
The number of transactions that fail depends on the rate of incoming transactions and the length of time needed to complete each transaction. Typically the number of failures is a very small percentage. This behavior is expected during an upgrade. If the failure level is not acceptable, schedule the upgrade during an off-hours maintenance window.

Note also that DataPower Gateway supports long-lived connections such as GraphQL subscriptions or other websockets connections. These long-lived connections might not be preserved when upgrading. Workloads with long-lived connections are more vulnerable to failed API transactions during upgrading.

You can limit the number of failed API transactions during the upgrade by using the DataPower Operator's `lifecycle` property to configure the `preStop` container lifecycle hook on the gateway pods. This approach mitigates the risk of API failures during the rolling update of the gateway StatefulSet by sleeping the pod for a span of time, allowing in-flight transactions to complete prior to the SIGTERM being delivered to the container. While this feature does not guarantee that no in-flight APIs would fail, it does provide some mitigation for in-flight transactions that can complete successfully within the configured time window. For more information, see [Delaying SIGTERM with preStop](#) in the DataPower documentation.

5. When upgrading a high-availability cluster, ensure that you meet the following requirements:
- Gateways must be updated one at a time.
 - Before starting the upgrade, a single gateway must be running as primary for all gateway-peering definitions.
 - When upgrading multiple gateways, the primary gateway must be upgraded last.
 - Ensure that the pod with a name like `gww6-0` or `gww5-0` is the primary because it is the last node to be upgraded.

To determine which gateway is running as primary, use either the `show gateway-peering-status` command in the DataPower CLI, or use the Gateway Peering Status display in the WebGUI in the API Connect application domain by

running the following command:

```
kubectl attach -it <podname>
```

To move the primary to the DataPower on which you're currently working, you can issue the following command:

```
gateway-peering-switch-primary <peering-object-name>
```

command.

About this task

When upgrading to a new mod release, the version must be changed to the latest mod release version on each CR. This change will be picked up by the operator, and the operator will then start the upgrade.

Procedure

1. Update the gateway CR for the new version of API Connect.

- a. Run the following command to edit the CR:

```
kubectl -n $NAMESPACE edit gw <CLUSTERNAME>
```

where **CLUSTERNAME** is the name specified in the subsystem CR at installation time.

- b. Update the API Connect **version** in the CR:

For example:

```
version: 10.0.5.4
```

- c. If you are upgrading to a version of API Connect that requires a new license, update the license value now.

For example:

```
license: L-GVEN-GFUPVE
```

For the list of licenses, see [API Connect licenses](#).

- d. Delete any **template** or **dataPowerOverride** sections. You cannot perform an upgrade if the CR contains an override.

- e. Run the following command to save and apply the CR: **wq**

When you save the updated CR, the upgrade starts automatically.

Known issue: Webhook error for incorrect license.

If you did not update the license ID in the CR, then when you save your changes, the following webhook error might display:

```
admission webhook "vmanagementcluster.kb.io" denied the request:
ManagementCluster.management.apiconnect.ibm.com "management" is invalid:
spec.license.license: Invalid value: "L-RJON-BYGHM4":
License L-RJON-BYGHM4 is invalid for the chosen version 10.0.5.4.
Please refer license document https://ibm.biz/apiclicenses
```

To resolve the error, see [API Connect licenses](#) for the list of the available license IDs and select the appropriate License IDs for your deployment. Update the CR with the new license value as in the following example, and then save and apply your changes again.

2. Run the following command to verify that the upgrade was successful:

```
kubectl get apic -n <namespace>
```

Example output after upgrading the gateway subsystem:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
gatewaycluster.gateway.apiconnect.ibm.com/gw1	2/2	Running	<version>	<version-build>	100m	

3. Optional: **If upgrading to v10.0.5.3 (or later) from an earlier 10.0.5.x release:** Review and configure the new inter-subsystem communication features: [Optional post-upgrade steps for upgrade to 10.0.5.3 \(or later\) from earlier 10.0.5 release](#).

Enabling gateway peering and verifying cluster status

To complete the operand upgrade, restart the gateway pods to enable gateway peering.

About this task

Follow these steps to fix gateway peering. Due to the ingress issuer changes, the gateway pods must be scaled down and back up. This process will cause 5 to 10 minutes of API downtime.

Procedure

1. After operand upgrade completes, wait for Management, Portal and Analytics subsystems to report **Running** status.
2. Scale down the gateway firmware containers by editing the Gateway CR and setting **replicaCount** to 0:

```
kubectl edit gw <gw-cr-name>
```

For example:

```
...
spec:
  replicaCount: 0
...
```

- Wait for the Gateway firmware pods to scale down and terminate. Ensure that the Gateway firmware pods are terminated before moving to next step.
- Scale up the Gateway firmware containers back to their original value, or remove the `replicaCount` field if none was there before.

```
kubectl edit gw <gw-cr-name>
```

- Your operands should all have an upgraded version. To verify, wait for the cluster status to become `Running`, and Reconciled version to show the new version number.

For example:

```
kubectl get apic -n <namespace>
```

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
analyticscluster.analytics.apiconnect.example.com/analytics	8/8	Running	10.0.4.0	10.0.4.0-1074		121m
NAME	PHASE	READY	SUMMARY	VERSION	AGE	
datapowerservice.datapower.example.com/gw1	Running	True	StatefulSet replicas ready: 1/1	10.0.4.0		100m
NAME	PHASE	LAST EVENT	WORK PENDING	WORK IN-PROGRESS	AGE	
datapowermonitor.datapower.example.com/gw1	Running		false	false		100m
NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
gatewaycluster.gateway.apiconnect.example.com/gw1	2/2	Running	10.0.4.0	10.0.4.0-1074		100m
NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
managementcluster.management.apiconnect.example.com/m1	16/16	Running	10.0.4.0	10.0.4.0-1074		162m
NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
portalcluster.portal.apiconnect.example.com/portal	3/3	Running	10.0.4.0	10.0.4.0-1074		139m

Note: Troubleshooting if Gateway pods not in sync with Management after upgrade

A rare failure has been seen during upgrade where some API Connect manager tasks are unable to run after the upgrade. Check if the API Connect 'taskmanager' pods log has the following error message. It would start 15 minutes after upgrade and repeat every 15 minutes for any stuck task.

```
TASK: Stale claimed task set to errored state:
```

If these errors are reported, restart all the `management-natscluster` pods, for example: `management-natscluster-1`.

```
kubectl -n <namespace> delete pod management-natscluster-1 management-natscluster-2 management-natscluster-3
```

Enabling Analytics after upgrading

After the upgrade successfully completes, re-enable Analytics data collection.

Before you begin

If upgrading from v10.0.5 you do not need to do these steps.

About this task

This task is only required if you are upgrading from releases prior to 10.0.5.0.

Procedure

- Re-create your analytics backup secret:
 - Delete the secret: `kubectl delete secret analytics-backup-secret`
 - Re-create the secret with a new name: `kubectl create secret generic <new backup secret name> --from-literal=access_key='Your_Access_Key' --from-literal=secret_key='Your_access_key_secret' -n Namespace_of_Analytics_Subsystem`
 - Update the credentials value in your analytics CR with the new backup secret name. Also, change your backup path to prevent the backups list showing pre-upgrade backups that you cannot restore from. For example:

```
spec:
  analytics:
    databaseBackup:
      ...
      credentials: <new backup secret name>
      path: <new backup path>
      ...
```

If this step is skipped, then backups will not run.

- Delete the old analytics PVCs. If this step is not done, then your previous (and now inaccessible) analytics data will continue to occupy storage space. List your PVCs and identify those that contain the following strings in their names: `storage-master`, `storage-coord`, `storage-data`, `storage-shared`, `kafka`, `zk`. These PVCs should all have an `AGE` that dates back to before you started your upgrade.



```
kubectl get pvc -n <namespace>
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
...						

data-ger2-analytics-storage-0	Bound	a7s-pv-8735e017	250Gi	RWO	rook-ceph-block	3h29m
data-ger2-analytics-storage-shared-0	Bound	a7s-pv-30aac7ed	250Gi	RWO	rook-ceph-block	227d
...						

In this example, `data-ger2-analytics-storage-shared-0` contains the string `storage-shared`, and it is 227 days old. To delete it, run:

```
kubectl delete pvc data-ger2-analytics-storage-shared-0 -n <namespace>
```

3. Add the new analytics:view permission to the Administrator role in the Cloud Manager UI.
 - a. Click , then select Roles
 - b. Click  on the Administrator role and select Edit
 - c. Select the checkbox next to the analytics permission and click save:

Analytics

View

4. Register new analytics services: [Registering an analytics service](#)
5. Associate your new analytics services with your gateways [Associating an analytics service with a gateway service](#)

Optional post-upgrade steps for upgrade to 10.0.5.3 (or later) from earlier 10.0.5 release

Version 10.0.5.3 introduces some new features that are related to inter-subsystem communication. This topic explains how to configure these features when you upgrade from a previous 10.0.5 release to v10.0.5.3 or later.

Enabling management CA certification on the REST API

The portal and gateway subsystems make requests to the management subsystem through the REST API that is hosted by the management subsystem. This communication is standard TLS, where each REST API endpoint has a server certificate that is presented to the client during TLS handshaking. For additional security, validation of the management server certificate CA is enabled on all subsystems for new installations of API Connect v10.0.5.3 and later. If you are upgrading from an earlier release, and want to enable this feature for the portal and gateway then edit the portal and gateway CRs, and add the following settings to the `spec`: section depending on if you are using `external` or `in-cluster` inter-subsystem communications ([in-cluster service communication between subsystems](#)):

- If you are using `external` inter-subsystem communication, in the portal CR add:

```
spec:
  mgmtPlatformEndpointCASecret:
    secretName: ingress-ca # Or <instance-name>-ingress-ca if using top-level CR. If you have customized certs, change
    ingress-ca to $PLATFORM_CA_SECRET
  mgmtConsumerEndpointCASecret:
    secretName: ingress-ca # Or <instance-name>-ingress-ca if using top-level CR. If you have customized certs, change
    ingress-ca to $CONSUMER_CA_SECRET
```

and in the gateway CR add:

```
spec:
  mgmtPlatformEndpointCASecret:
    secretName: ingress-ca # Or <instance-name>-ingress-ca if using top-level CR. If you have customized certs, change
    ingress-ca to $PLATFORM_CA_SECRET
```

where:

- `$PLATFORM_CA_SECRET` is the Kubernetes secret object that contains the CA certificate that is used by the platform REST API.
 - `$CONSUMER_CA_SECRET` is the Kubernetes secret object that contains the CA certificate that is used by the consumer REST API.
- If you are using `in-cluster` inter-subsystem communications, in the portal CR add:

```
spec:
  mgmtPlatformEndpointSvcCASecret:
    secretName: management-ca # Or <instance-name>-mgmt-ca if using top-level CR. If you have customized certs, change
    ingress-ca to $PLATFORM_CA_SECRET
  mgmtConsumerEndpointSvcCASecret:
    secretName: management-ca # Or <instance-name>-mgmt-ca if using top-level CR. If you have customized certs, change
    ingress-ca to $CONSUMER_CA_SECRET
```

and in the gateway CR add:

```
spec:
  mgmtPlatformEndpointSvcCASecret:
    secretName: management-ca # Or mgmt-ca or <instance-name>-mgmt-ca if using top-level CR. If you have customized certs,
    change ingress-ca to $PLATFORM_CA_SECRET
```

If changing to use `in-cluster` inter-subsystem communication after an upgrade, you must also reregister all subsystems: [Converting existing registered subsystems to use in-cluster communications](#).

Note: If you are not sure of the secret names, follow these steps:

1. Check the management subsystem CR with `kubectl describe mgmt`, and identify the issuer of the platform and consumer API endpoints:

```
kubectl describe mgmt -n <namespace>
```

```
...
Platform API Endpoint:
Annotations:
```

```

cert-manager.io/issuer: ingress-issuer
...
Consumer API Endpoint:
Annotations:
cert-manager.io/issuer: ingress-issuer

```

- Describe the issuer with `kubectl describe issuer`, and identify the secret name:

```

kubectl describe issuer ingress-issuer -n <namespace>
...
Spec:
Ca:
Secret Name: ingress-ca
...

```

Enable JWT security for inter-subsystem communication

If you want to use JWT security instead of mTLS for inter-subsystem communication, then follow these steps. For more information on JWT security, see: [Enable JWT security instead of mTLS](#).

- Describe the `mgmt` CR to get the JWKS URL:

```

kubectl describe mgmt -n <namespace>

status:
- name: jwksUrl
  secretName: api-endpoint
  type: API
  uri: https://api.apic.acme.com/api/cloud/oauth2/certs

```

- For each portal, gateway, and analytics subsystem where you want to use JWT security, edit the CR corresponding that subsystem to disable mTLS and specify the `jwksUrl`:

```

spec:
...
mtlsValidateClient: false
jwksUrl: <JWKS URL>

```

<JWKS URL> is the URL identified in step 1.

- v10.0.5.4 and later:** Enable JWT on the gateway to analytics communications flow. Enable the Use JWT switch for the registered gateway in the Topology page of the Cloud Manager UI.

Datapower version 10.5.0.4 only: To enable JWT on the gateway to analytics communications flow, you also must add a `dataPowerOverride` section to the gateway CR:

```

spec:
jwksUrl: <JWKS URL>
dataPowerOverride:
  image: customregistry.com/custom-image-datapower:10.5.0.5
  version: 10.5.0.5
  license: X-XXXX-XXXXXX

```

Converting existing registered subsystems to use in-cluster communications

If you want to change your existing portal, gateway, and analytics subsystems to use `in-cluster` inter-subsystem communication then they must be reregistered. For more information about `in-cluster` communication, see [In-cluster service communication between subsystems](#).

Important: Understand the following points before you change the communication type of a subsystem:

- Reregistering any of the subsystems requires an outage of that subsystem.
- When you change communication type, any backup of the management subsystem that was taken before this change becomes unusable. Do not change the inter-subsystem communication type if you might want to restore the management subsystem from an earlier backup.
- To reregister the portal, all portal sites must be deleted first. Deleting a portal site means that any customizations that were made to the site must manually be reapplied after the portal is reregistered and the site recreated.
- To reregister the gateway, all published products must be deleted from the gateway.

To reregister a portal subsystem, the steps are:

- Delete all portal sites from the portal service. From the API Manager UI, for each catalog that has a site on the portal, delete the portal site: [Edit or delete portal site](#).
- Delete the portal service in the Topology section of the Cloud Manager UI.
- Reregister the portal service, specifying the inter-subsystem communication type that you want, in the Topology section of the Cloud Manager UI. See [Register portal service](#).

To reregister a gateway subsystem, the steps are:

- In the API Manager UI, delete all products from all catalogs that use the gateway service. See [Removing a product from a catalog](#).
- In the API Manager UI, for each catalog that uses the gateway service, remove the gateway service from the catalog settings. See [Configure catalog gateway service](#).
- Delete the gateway service in the Topology section of the Cloud Manager UI.
- Reregister the gateway service, specifying the inter-subsystem communication type that you want, in the Topology section of the Cloud Manager UI. See [Registering a gateway service](#).

To reregister the analytics subsystem, the steps are:

- Unassociate the analytics service from all gateway services that it is associated with. See [Unassociate analytics service](#).
- Delete the analytics service in the Topology section of the Cloud Manager UI.

3. Reregister the analytics service, specifying the inter-subsystem communication type that you want, in the Topology section of the Cloud Manager UI. See [Registering an analytics service](#).

Upgrading from 2018 on native Kubernetes

You can upgrade from the latest release of v2018 to the latest release of v10.

About this task

Complete the API Connect upgrade from v2018 to v10 by performing the following tasks:

- [Preparing the 2018 deployment for upgrade](#)
Prepare your 2018 deployment for upgrade to 10.0.5.x.
- [Obtaining files for upgrade from 2018](#)
Obtain the product files, load them into a local Docker registry, and then upload them as required to a remote Docker registry. Decompress the release files that are distributed as a zip file.
- [Deploying operators in a single-namespace API Connect cluster](#)
When upgrading from 2018 to the latest release of v10, first deploy the operators to your namespace.
- [Deploying operators in a multi-namespace API Connect cluster](#)
When you have a multiple namespace deployment in 2018, use these instructions to deploy the version 10 operators.
- [Upgrading the Management subsystem from 2018 to v10](#)
You can upgrade the management subsystem from 2018 to the latest version of v10.
- [Upgrading the Developer Portal subsystem from 2018 to v10](#)
You can upgrade the Developer Portal subsystem from 2018.4.1.20 or later to the latest release of v10.
- [Upgrading the Analytics subsystem from v2018 to v10](#)
It is not possible to directly upgrade the analytics subsystem from v2018 to v10.0.5. New v10.0.5 analytics subsystems must be deployed to replace your v2018 analytics subsystems.
- [Upgrading the gateway subsystem from 2018 to v10](#)
You can upgrade the DataPower Gateway subsystem from API Connect 2018.4.1.20 or later to the latest release of v10.
- [Post-upgrade steps for upgrading to v10](#)
After you upgrade API Connect from v2018 to v10, complete these final steps.
- [Disaster recovery of a 2018 deployment](#)
You can optionally return to a 2018 deployment, if necessary, after completion of a successful upgrade from 2018 to v10.

Preparing the 2018 deployment for upgrade

Prepare your 2018 deployment for upgrade to 10.0.5.x.

Before you begin

Before you move from 2018 to 10.0.5.x, ensure that you have reviewed v10 requirements and features:

- Review the IBM Support page [API Connect: v2018 to v10 upgrade paths](#) to determine whether your Version 2018 deployment can be directly upgraded to the latest release of API Connect.
- [IBM API Connect Version 10 software product compatibility requirements](#)
- [Deployment requirements](#)
- [Planning your deployment](#)
- [What's New in the latest release](#)
- [Version 10 known limitations](#)

About this task

Before you start an upgrade, make sure that your 2018 database is in a consistent state, perform the necessary backups, and set up external logging.

Procedure

1. Review the upgrade considerations. See [Upgrade considerations on native Kubernetes](#).

2. Configure external logging prior to upgrade.

This step is not mandatory but is recommended. For example, you could use <https://logdna.com/>.

3. Be sure you know the name of each subsystem on your 2018 deployment.
The 2018 subsystem names are carried over to version 10 installations. To view the names:

```
apicup subsys list
```

These subsystem names are the values used in the `spec.subsystemName` field for each custom resource (CR) when upgrading.

4. On 2018, check the health of each subsystem before upgrade:

```
apicup subsys health-check <subsystem_name>
```

5. On the 2018 system, back up the original `apicup` project directory to a location from which it can always be retrieved.

Note: The original project directory created with the `apicup` installer during the initial 2018 product installation (for example, `myProject`) is required if you need to restore the database and to run disaster recovery back to v2018. You cannot restore the database or perform a disaster recovery without the initial project directory

- because it contains pertinent information about the 2018 cluster.
- Complete a full backup for the API Connect 2018 Management, Portal, and Analytics subsystems. Verify that you can successfully restore from the backups.
Note: The Analytics subsystem cannot be upgraded due to changes in v10.0.5.0, and you cannot restore it into v10 but you should back it up in case a rollback is needed.
Make sure you have current and correct configuration for backup and restore for all of the 2018 subsystems. See [Backing up and restoring](#).
 - If you want to retain your analytics data, then you must export it before running the upgrade.
For instructions on exporting data from v2018, see the v2018 documentation: <https://www.ibm.com/docs/en/api-connect/2018.x?topic=analytics-exporting-api-event-data>. Also, you can set up analytics offload so that all new analytics data is sent to a third-party system for storage: <https://www.ibm.com/docs/en/api-connect/2018.x?topic=cloud-configuring-analytics-offload-api-connect>.
 - Disassociate and delete your Analytics services.
Note: If you want your v2018 deployment to continue gathering analytics data up to the last possible moment, this step can instead be done just before the Management subsystem upgrade: [5](#).
 - In Cloud Manager, click Topology.
 - In the section for the Availability Zone that contains the Analytics service, locate the Gateway service that the Analytics service is associated with.
 - Click the actions menu, and select Unassociate analytics service. Remember to disassociate each Analytics service from all Gateways.
 - In the section for the Availability Zone that contains the Analytics services, locate each Analytics service and click Delete.
 - On the 2018 deployment, use the `apicops` utility to resolve any known inconsistencies and synchronization problems with the database:
The `apicops preupgrade` command must be run in fix mode just prior to starting the management upgrade. The fix mode fixes any inconsistencies and synchronization problems in the database.

Important: Upgrade will fail if the `apicops` command is not successfully completed within 2 hours of starting the management upgrade. If the management upgrade must be run more than once, you must run the command in fix mode each time within 2 hours of starting the management upgrade.
 - Obtain the latest 2018 release of `apicops` from <https://github.com/ibm-apiconnect/apicops#latest-v2018-release>.
 - Set up the environment. See <https://github.com/ibm-apiconnect/apicops#requirements>.
 - Run the command:

```
apicops preupgrade -m fix --upgrade_to v10 -l <outputLogFileName>
```

 - The `-m` option runs the utility in fix mode. Running the utility in fix mode resolves most problems. If the utility identifies a problem it cannot resolve, contact IBM Support.
 - Set the `-l` option to send output to a log file.
10. Next, install the upgrade files. Continue with [Obtaining files for upgrade from 2018](#).

Obtaining files for upgrade from 2018

Obtain the product files, load them into a local Docker registry, and then upload them as required to a remote Docker registry. Decompress the release files that are distributed as a zip file.

Before you begin

Verify you completed [Preparing the 2018 deployment for upgrade](#).

About this task

To upgrade from v2018, complete the steps in this task to navigate to [IBM Fix Central](#) site and download the Docker image-tool file of the API Connect subsystems, as well as the Kubernetes operators and API Connect Custom Resource (CR) templates. Next, upload the image-tool file to your Docker local registry. If necessary, you can populate a remote container registry with repositories. Then you can push the images from the local registry to the remote registry. Decompress the zip files that contain the operators and templates, so that you can use those files during deployment.

Procedure

- Obtain the API Connect files from IBM Fix Central:
The following files are used for deployment on native Kubernetes:
 - IBM® API Connect <version> for Containers**
Docker images for all API Connect subsystems
 - IBM® API Connect <version> Operator Release Files for Containers
Kubernetes operators and API Connect Custom Resource (CR) templates
 - IBM® API Connect <version> Operator Upgrade Release Files for Containers
Kubernetes upgrade operators and API Connect upgrade CRs.
 - IBM® API Connect <version> Toolkit for <operating_system_type>
Toolkit command line utility. Packaged standalone, or with API Designer or Loopback:
 - IBM® API Connect <version> Toolkit for <operating_system_type>
 - IBM® API Connect <version> Toolkit with Loopback for <operating_system_type>
 - IBM® API Connect <version> Toolkit Designer with Loopback for <operating_system_type>
 Not required during initial installation. After installation, you can download directly from the Cloud Manager UI and API Manager UI. See [Installing the toolkit](#).
 - IBM® API Connect <version> Local Test Environment
Optional test environment. See [Testing an API with the Local Test Environment](#)
 - IBM® API Connect <version> Security Signature Bundle File
Checksum files that you can use to verify the integrity of your downloads.

- Load the image-tool image in your Docker local registry:

```
docker load < apiconnect-image-tool-<version>.tar.gz
```

Ensure that the registry has sufficient disk space for the files.

3. If your Docker registry requires repositories to be created before images can be pushed, create the repositories for each of the images listed by the image tool. (If your Docker registry does not require creation of repositories, skip this step and go to Step 4.)

- a. Run the following command to get a list of the images from image-tool:

```
docker run --rm apiconnect-image-tool-<version> version --images
```

For `docker run`: For example, version is 10.0.1.5. Note that the value does not contain an `-eus` suffix.

- b. From the output of each entry of the form `<image-name>:<image-tag>`, use your Docker registry repository creation command to create a repository for `<image-name>`.

For example in the case of AWS ECR the command would be for each `<image-name>`:

```
aws ecr create-repository --repository-name <image-name>
```

4. Upload the image:

- If you do not need to authenticate with the docker registry, use:

```
docker run --rm apiconnect-image-tool-<version> upload <registry-url>
```

- Otherwise, if your docker registry accepts authentication with username and password arguments, use:

```
docker run --rm apiconnect-image-tool-<version> upload <registry-url> --username <username> --password <password>
```

- Otherwise, such as with IBM Container Registry, if you need the image-tool to use your local Docker credentials, first authenticate with your Docker registry, then upload images with the command:

```
docker run --rm -v ~/.docker:/root/.docker --user 0 apiconnect-image-tool-<version> upload <registry-url>
```

Note: The previous command does not work on macOS if Docker is configured to use the `osxkeychain` credential store. In this case, complete the following steps:

- Disable **Docker > Preferences... > Securely store Docker logins in the macOS keychain**.
- Inspect `~/.docker/config.json` to make sure that it does not contain `"credStore": "osxkeychain"`, as some versions of Docker-for-mac may handle the setting correctly per <https://github.com/docker/for-mac/issues/4192>.
- Authenticate with the Docker registry you intend to upload to.
- Run the command:

```
docker run --rm -v ~/.docker:/root/.docker --user 0 apiconnect-image-tool-<version> upload <registry-url>
```

- Once the upload of images is successful you may enable **Docker > Preferences... > Securely store Docker logins in the macOS keychain**.

Docker authentication notes:

- Both HTTPS and HTTP are supported. Best practice for Docker registry security is to use HTTPS by utilizing standard Docker load tools, to ensure your images and platform are protected. However, when necessary you can use HTTP by specifying the `--tls-verify=false` flag.
- When using a Docker registry on `localhost`, you might encounter that the image-tool Docker container is on the Docker network and typically without access to `localhost`. You can work around this issue by using the `--network host` argument to the Docker command. Note that `--network host` is a Docker argument, not an image-tool argument.
- Example of using `--network host` and `--tls-verify=false`:

```
$ docker run -d -p 5000:5000 --name registry registry:2
...
$ docker run --rm --network host apiconnect-image-tool-<version> upload localhost:5000 --tls-verify=false
...
```

Providing a certificate for verification, or disabling TLS verification

See the sample output of the tool usage for options to provide a certificate for verification or to disable TLS verification:

```
$ docker run --rm apiconnect-image-tool-<version> upload --help
upload docker images
```

Usage:

```
image-tool upload REGISTRY [flags]
```

Flags:

```
--cert-dir string  Directory with destination registry certificate tls.crt file
--username string  User name
-h, --help         help for upload
--password         password for <username>
--tls-verify       Verify TLS on destination registry (default true)
```

Global Flags:

```
--accept-license  Accept the license for API Connect
--debug           Enable debug logging
```

Notes:

- `--username=<username>` and `--password=<password>` can be used to specify credentials for authentication with the destination Docker registry.
- `--tls-verify=false` can be used to disable verification of the destination Docker registry certificate
- `--cert-dir <path>` can be used to provide a `tls.crt` file to be used for validation of the destination Docker registry certificate. For example:

```
docker run --rm -v <path-to-folder-with-tls.crt-file>:/cert apiconnect-image-tool-<version> upload <registry> --cert-dir /cert
```

5. Download and decompress the IBM® API Connect `<version>` files needed for upgrade:

- a. Decompress IBM® API Connect `<version>` Operator Release Files for Containers.

Contents:

- API Connect Operator operator custom resource definition (CRDs)
- API Connect Operator Deployment and required resources CRDs
- API Connect Operator Deployment and required resources CRDs, for multiple-namespace installations
- DataPower Gateway Operator operator custom CRDs
- API Connect custom resource templates, and Certificate Manager

The Operator Release for Containers files will be used later in deployment instructions.

Note: The Operator Release files may include catalog and operator source for OpenShift. These files are not used when upgrading on native Kubernetes.

- b. Decompress IBM® API Connect <version> Operator Upgrade Release Files for Containers.

Contents:

- API Connect Operator upgrade operator custom resource definition (CRDs).
- API Connect Upgrade Operator Deployment and required resources CRDs, for single-namespace installations.
- API Connect Upgrade Operator Deployment and required resources CRDs, for multiple-namespace installations.
- API Connect upgrade custom resource templates, and Certificate Manager.

- c. Decompress the archive `helper_files.zip` from IBM® API Connect <version> Operator Upgrade Release Files for Containers, to access the API Connect custom resource templates and Certificate Manager for use in upgrade.

The zip file contains custom resource templates for:

- Upgrade CRs for each subsystem (Management, Portal, Gateway, and Analytics)
- Certificate Manager
- Ingress Issuer and Subsystem Certificates Resources.

The templates will be used later in deployment instructions.

6. Next, prepare the API Connect cluster for upgrade.

There are separate instructions for 2018 single namespace clusters and multi-namespace cluster. Continue with the instructions that apply to your deployment:

- [Deploying operators in a single-namespace API Connect cluster](#)
- [Deploying operators in a multi-namespace API Connect cluster](#)

Deploying operators in a single-namespace API Connect cluster

When upgrading from 2018 to the latest release of v10, first deploy the operators to your namespace.

Before you begin

You must have completed the prerequisite steps and obtained the latest API Connect v10 software. See [Preparing the 2018 deployment for upgrade](#).

Procedure

1. Apply the new version 10 Custom Resource Definitions (CRDs) and the upgrade operator CRDs into the 2018 cluster at your namespace.
 - a. version 10 CRDs:

```
kubectl apply -f ibm-apiconnect-crds.yaml
```

- b. version 2018 to version 10 upgrade CRDs:

```
kubectl apply -f ibm-apiconnect-upgrade-crds.yaml
```

2. Create secrets to pull down version 10 images from docker registry:

- a. Create `apic-registry-secret` secret to pull down product v10 API Connect images.

```
kubectl create secret docker-registry apic-registry-secret
--docker-server=<registry_server>
--docker-username=<username@example.com>
--docker-password=<password>
--docker-email=<username@example.com>
-n <v2018_namespace>
```

- b. Create `datapower-docker-local-cred` secret to pull down DataPower images:

```
kubectl create secret docker-registry datapower-docker-local-cred
--docker-server=<docker_server>
--docker-username=<username@example.com>
--docker-password=<password>
--docker-email=<username@example.com>
-n <v2018_namespace>
```

- c. Create `datapower-admin-secret`

```
kubectl create secret generic datapower-admin-credentials
--from-literal=password=admin
-n <v2018_namespace>
```

You will use the image pull secret key name in the CR files, `spec.ImagePullSecret`.

3. Install cert-manager and configure certificates:

- a. Install cert-manager 1.7.1 but do not specify a custom namespace because cert manager will be installed in its own namespace.

```
kubectl apply -f <path>/cert-manager-1.9.1.yaml
```

- b. Wait for cert-manager pods to enter `Running 1/1` status before proceeding. Use the following command to check:

```
kubectl -n cert-manager get po
```

There are 3 cert-manager pods in total.

4. Install `ingress-issuer-v1.yaml`.

```
kubectl apply -f <path>/ingress-issuer-v1.yaml -n <v2018_namespace>
```

This step creates ingress resources in the `namespace` of v2018 subsystems.

5. Apply the version 10 apiconnect-operator into your namespace.

The apiconnect-operator launches and maintains the v10 micro-services.

- If you are using a namespace other than `default`, open `ibm-apiconnect.yaml` in a text editor. Replace each occurrence of `default` with the 2018 namespace for your deployment.
- Open `ibm-apiconnect.yaml` in a text editor. Replace `REPLACE-DOCKER-REGISTRY` with the root of the docker registry you are working with.
- Install the ibm-apiconnect Kubernetes deployment.

```
kubectl apply -f ibm-apiconnect.yaml -n <2018_namespace>
```

This command gets the version 10 apiconnect-operator up and running in the namespace of the 2018 subsystems.

6. Start the DataPower operator in the cluster namespace.

Note: Deployment of the DataPower operator is only needed if you have at least one API Connect 2018 Gateway subsystem (whether v5-compatible or not) to upgrade. If you are using DataPower in a different form factor such as an Appliance, you will not have an API Connect 2018 Gateway subsystem to upgrade, and will not need the DataPower operator for your upgrade.

Create `ibm-datapower.yaml` in the cluster namespace:

- Open `ibm-datapower.yaml` and replace `REPLACE-DOCKER-REGISTRY` with the root of the docker registry you are working with.
- Use the following command to start the version 10 operator in the cluster (namespace):

```
kubectl apply -f ibm-datapower.yaml -n <2018_namespace>
```

7. Apply the version 10 ibm-apiconnect-upgrade operator into your namespace, to perform the 2018-to-10 upgrade process.

- Open `ibm-apiconnect-upgrade.yaml` in a text editor.
 - If you are using a namespace other than `default`, replace each occurrence of `default` with the API Connect 2018 namespace for your deployment.
 - Replace `REPLACE-DOCKER-REGISTRY` with the root of the docker registry you are working with.
 - Optionally, you can supply values for the environment variables `DEFAULT_IMAGE_REGISTRY` and `DEFAULT_IMAGE_PULL_SECRET` in `ibm-apiconnect-upgrade.yaml`
If you do not set these variables in `ibm-apiconnect-upgrade.yaml`, you can (and must) set `$IMAGE_REGISTRY` and `$IMAGE_PULL_SECRET` later in the upgrade CR for each subsystem.

Example:

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ibm-apiconnect-upgrade
<...>
containers:
- name: ibm-apiconnect-upgrade
  image: REPLACE-DOCKER-REGISTRY/ibm-apiconnect-upgrade-
operator@sha256:b1e09c3ed2f52b6724c7ef8a399ac36db63f9f64f19458736fec9a043458435e
  command:
  - ibm-apiconnect-upgrade
  args:
  - --zap-time-encoding=iso8601
  imagePullPolicy: Always
  env:
  - name: WATCH_NAMESPACE
    valueFrom:
      fieldRef:
        fieldPath: metadata.namespace
  - name: POD_NAMESPACE
    valueFrom:
      fieldRef:
        fieldPath: metadata.namespace
  - name: POD_NAME
    valueFrom:
      fieldRef:
        fieldPath: metadata.name
  - name: OPERATOR_NAME
    value: ibm-apiconnect-upgrade
  - name: DEFAULT_IMAGE_REGISTRY
    value: <registry-url>
  - name: DEFAULT_IMAGE_PULL_SECRET
    value: <image-pull-secret-name>
```

- Install the ibm-apiconnect upgrade Kubernetes deployment.

```
kubectl apply -f ibm-apiconnect-upgrade.yaml -n <v2018_namespace>
```

This command gets the v10 apiconnect upgrade operator up and running in the namespace.

8. Next, upgrade the subsystems. Continue with [Upgrading the Management subsystem from 2018 to v10](#).

Deploying operators in a multi-namespace API Connect cluster

When you have a multiple namespace deployment in 2018, use these instructions to deploy the version 10 operators.

Before you begin

You must have completed the prerequisite steps and obtained the latest API Connect v10 software. See [Preparing the 2018 deployment for upgrade](#) and [Obtaining files for upgrade from 2018](#).

Procedure

1. Prepare your environment:

a. Ensure KUBECONFIG is set for the target cluster:

```
export KUBECONFIG=<path_to_cluster_config_YAML_file>
```

An example path is `/Users/user/.kube/clusters/<cluster_name>/kube-config-<cluster_name>.yaml`

b. Create namespaces on which subsystems are planned to be installed.

The required namespaces are:

- Apiconnect operator namespace
- Gateway subsystem namespace
- Management subsystem namespace
- Portal subsystem namespace
- Analytics subsystem namespace

The following example values and deployment are used in this topic:

- Apiconnect operator is installed in `operator` namespace
- DataPower operator and gateway subsystem is installed in `gtw` namespace
- Management subsystem is installed in `mgmt` namespace
- Portal subsystem is installed in `portal` namespace
- Analytics subsystem is installed in `a7s` namespace

c. Multi-namespace templates are provided in `helper_files.zip`, inside the `release_files.zip` you downloaded in [Obtaining files for upgrade from 2018](#):

```
helper_files_unpack/multi-ns/
```

2. Install cert-manager and configure certificates:

a. Install cert-manager 1.7.1 but do not specify a custom namespace because cert manager will be installed in its own namespace.

```
kubectl apply -f cert-manager-1.9.1.yaml
```

b. Wait for cert-manager pods to enter `Running 1/1` status before proceeding. Use the following command to check:

```
kubectl -n cert-manager get po
```

There are 3 cert-manager pods in total.

c. Install `ingress-ca` certificate:

- Locate `ingress-ca` certificate at `helper_files/multi-ns/ingress-ca-cert.yaml`
- Create `ingress-ca` certificate in one of the subsystems. For example, if we choose `mgmt` namespace:

```
kubectl -n mgmt apply -f ingress-ca-cert.yaml
```

d. Install common issuers on all the namespaces:

- Locate `common-issuer.yaml` in `helper_files/multi-ns-support/`
- Create `common-issuer.yaml` in all namespaces:

```
kubectl create -f common-issuer.yaml -n mgmt
kubectl create -f common-issuer.yaml -n gtw
kubectl create -f common-issuer.yaml -n ptl
kubectl create -f common-issuer.yaml -n a7s
```

e. Obtain the ingress-ca secret created by cert manager and create the secret on all namespaces:

- Make sure ingress-ca certificate `READY` status is set to true:

```
kubectl get certificate ingress-ca -n mgmt
NAME      READY  SECRET  AGE
ingress-ca  True   ingress-ca  9m24s
```

- Cert-manager creates a secret called `ingress-ca` in `mgmt` namespace which represents the certificate we created in Step [2.d](#).

- Use the following command to obtain the yaml format of the secret:

```
kubectl get secret ingress-ca -n <namespace-where-ingress-ca-cert-is-created> -o yaml > ingress-ca-secret-in-cluster.yaml
```

- Remove unwanted content from the secret yaml:

```
cat ingress-ca-secret-in-cluster.yaml | grep -v 'creationTimestamp' | grep -v 'namespace' | grep -v 'uid' | grep -v 'resourceVersion' > ingress-ca-secret.yaml
```

- Create the secret using the yaml we prepared in previous step on rest of the subsystem namespaces. In this example, in the `gtw`, `a7s` and `ptl` namespaces.

```
kubectl create -f ingress-ca-secret.yaml -n gtw
kubectl create -f ingress-ca-secret.yaml -n ptl
kubectl create -f ingress-ca-secret.yaml -n a7s
```

- Install management certs in management namespace:

- i. Locate `management-certs.yaml` in `helper_files/multi-ns-support/`
- ii. Create `management-certs.yaml` in the `mgmt` namespace:

```
kubectl create -f management-certs.yaml -n mgmt
```

- g. Install gateway certs in the gateway namespace:
 - i. Locate `gateway-certs.yaml` in `helper_files/multi-ns-support/`
 - ii. Create `gateway-certs.yaml` in the `gtw` namespace:

```
kubectl create -f gateway-certs.yaml -n gtw
```

3. Install the apiconnect operator:

- a. Install the `ibm-apiconnect` CRDs with the following command. Do not specify a namespace:

```
kubectl apply -f ibm-apiconnect-crds.yaml
```

- b. Create a registry secret with the credentials to be used to pull down product images with the following command, replacing `<v10_Operator_Namespace>` with the desired namespace for the apiconnect operator deployment.

```
kubectl -n <v10_Operator_Namespace> create secret docker-registry apic-registry-secret
--docker-server=apic-dev-docker-local.artifactory.swg-devops.com
--docker-username=XXX@ibm.com --docker-password=XXXXXX
--docker-email=XXX@ibm.com
```

- For example, replace `<v10_Operator_Namespace>` with `operator`.
- `docker-password` can be your artifactory API key.
- `-n <v10_Operator_Namespace>` may be omitted if `default` namespace is being used for installation.

- c. Locate and open `ibm-apiconnect-distributed.yaml` in a text editor of choice. Then, find and replace each occurrence of `$OPERATOR_NAMESPACE` with `<v10_Operator_Namespace>`, replacing `<v10_Operator_Namespace>` with the desired namespace for the deployment. In this example, the namespace is `operator`.

- d. Also in `ibm-apiconnect-distributed.yaml`, locate the `image:` keys in the containers sections of the deployment yaml right below `imagePullSecrets:`. Replace the placeholder values `REPLACE-DOCKER-REGISTRY` of the `image:` keys with the docker registry host location of the apiconnect operator image (either uploaded to own registry or pulled from public registry).

- e. Install `ibm-apiconnect-distributed.yaml` with the following command

```
kubectl apply -f ibm-apiconnect.yaml -n <v10_Operator_Namespace>
```

4. Install DataPower operator:

Note: Deployment of the DataPower operator is only needed if you have at least one API Connect 2018 Gateway subsystem (whether v5-compatible or not) to upgrade. If you are using DataPower in a different form factor such as an Appliance, you will not have an API Connect 2018 Gateway subsystem to upgrade, and will not need the DataPower operator for your upgrade.

- a. Create a registry secret for the DataPower registry with the credentials to be used to pull down product images with the following command, replacing `<namespace>` with the desired namespace for the deployment. (In this case `gtw`):

```
kubectl -n <v2018_gateway_subsystem> create secret docker-registry datapower-docker-local-cred
--docker-server=datapower-docker-local.artifactory.swg-devops.com
--docker-username=XXX@ibm.com --docker-password=XXXXXX
--docker-email=XXX@ibm.com
```

- For example, replace `<v2018_gateway_subsystem>` with `gtw`.
- `docker-password` can be your artifactory API key.
- `-n <v2018_gateway_subsystem>` may be omitted if `default` namespace is being used for installation

- b. Create a DataPower admin secret, replacing `<v2018_gateway_subsystem>` with the desired namespace for the deployment. For example, `gtw`. This secret will be used for `$ADMIN_USER_SECRET` later in the Gateway CR:

```
kubectl -n <v2018_gateway_subsystem> create secret generic datapower-admin-credentials --from-literal=password=admin
```

`<v2018_gateway_subsystem>` may be omitted if `default` namespace is being used for installation.

- c. Locate and open `ibm-datapower.yaml` in a text editor of choice. Then, find and replace each occurrence of `default` with `<v2018_gateway_subsystem>`, replacing `<v2018_gateway_subsystem>` with the desired namespace for the deployment. For example, `gtw`.

- d. Install `ibm-datapower.yaml` with the following command:

```
kubectl -n <v2018_gateway_subsystem> apply -f ibm-datapower.yaml
```

5. Install the API Connect upgrade operator:

- a. Create upgrade CRDs:

```
kubectl apply -f <path>/ibm-apiconnect-upgrade-crds.yaml
```

- b. The `ibm-apiconnect-upgrade-operator` must be installed in the namespace of the `ibm-apiconnect-operator`.

- c. Open the `ibm-apiconnect-upgrade-distributed.yaml` in a text editor:

Note:

You can obtain `ibm-apiconnect-upgrade-distributed.yaml` from the file IBM® API Connect `<version>` Operator Upgrade Release Files for Containers that you downloaded in [Obtaining files for upgrade from 2018](#).

- i. Replace all the `$OPERATOR_NAMESPACE` namespaces with the namespace of `ibm-apiconnect-operator`.
- ii. Replace `REPLACE-DOCKER-REGISTRY` with the root of the docker registry you are working with.
- iii. Optionally, you can supply values for the environment variables `DEFAULT_IMAGE_REGISTRY` and `DEFAULT_IMAGE_PULL_SECRET` in `ibm-apiconnect-upgrade-distributed.yaml`

Example:

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ibm-apiconnect-upgrade
<...>
```

```

containers:
- name: ibm-apiconnect-upgrade
  image: REPLACE-DOCKER-REGISTRY/ibm-apiconnect-upgrade-
operator@sha256:b1e09c3ed2f52b6724c7ef8a399ac36db63f9f64f19458736fec9a043458435e
  command:
  - ibm-apiconnect-upgrade
  args:
  - --zap-time-encoding=iso8601
  imagePullPolicy: Always
  env:
  - name: WATCH_NAMESPACE
    valueFrom:
      fieldRef:
        fieldPath: metadata.namespace
  - name: POD_NAMESPACE
    valueFrom:
      fieldRef:
        fieldPath: metadata.namespace
  - name: POD_NAME
    valueFrom:
      fieldRef:
        fieldPath: metadata.name
  - name: OPERATOR_NAME
    value: ibm-apiconnect-upgrade
  - name: DEFAULT_IMAGE_REGISTRY
    value: <registry-url>
  - name: DEFAULT_IMAGE_PULL_SECRET
    value: <image-pull-secret-name>

```

d. Run the following command to get `ibm-apiconnect-upgrade-operator` up and running in the cluster:

```
kubectl apply -f ibm-apiconnect-upgrade-distributed.yaml -n <v10_Operator_Namespace>
```

`ibm-apiconnect-upgrade-operator` is used to perform the v2018-v10 upgrade process.

6. Next, upgrade the subsystems. Continue with [Upgrading the Management subsystem from 2018 to v10](#).

Note: Unless otherwise stated, when performing the individual subsystem upgrades the `<namespace>` value of example `kubectl` commands for a given subsystem should be set to the namespace for the particular subsystem component they are acting on.

Upgrading the Management subsystem from 2018 to v10

You can upgrade the management subsystem from 2018 to the latest version of v10.

Before you begin

Before you can upgrade the management subsystem, you must prepare the API Connect cluster namespaces. If you haven't done so already, see one of the following:

- [Deploying operators in a single-namespace API Connect cluster](#)
- [Deploying operators in a multi-namespace API Connect cluster](#)

About this task

- One of the major changes between 2018 and v10 is the change from Cassandra to Postgres for the management database. The upgrade process uses the following steps to upgrade the management subsystem database:
 1. Take management database backup.
Backup and restore must be configured for management subsystem on 2018.
 2. Scale down the 2018 pods except Cassandra database
 3. Extract data from Cassandra database.
This task runs as an `extract` job. The extract job creates logs.
 4. Bring up Postgres database.
 5. Load the extracted data into Postgres database and do the necessary transformations.
This task runs as the `load` job. The load job creates a log.
- To upgrade the management subsystem, you configure an upgrade CR and apply it. When you apply the CR, the upgrade process runs automatically.

Procedure

1. Take a full backup of the API Connect 2018 Management subsystem. This backup is in addition to the backup taken in [Preparing the 2018 deployment for upgrade](#), which was taken before the pre-upgrade operations. The pre-upgrade operations will have changed some of the contents of the database and so it is recommended to have backups from before and after.
2. Populate the template upgrade CR for `management.apiconnect.ibm.com_v1beta1_upgradefromv2018_cr.yaml` with values for your deployment:

Table 1. Variables to set in the Upgrade CR template

Variable	Description
<code>\$NAME</code>	Name to uniquely identify this resource. Example: <code>mgmt</code>

Variable	Description
<code>\$MANAGEMENT_SUBSYS TEM_NAME</code>	Must match the v2018's management subsystem name. Use <code>apicup subsys list</code> to find the name. For example: <pre>\$ apicup subsys list Name Type Target mgmt ManagementSubsystem kubernetes gwv5 GatewaySubsystem kubernetes gwv6 GatewaySubsystem kubernetes portal PortalSubsystem kubernetes analytics AnalyticsSubsystem kubernetes</pre>
<code>\$TRIGGER_ROLLBACK</code>	Can be <code>true</code> or <code>false</code> . Must initially be set to <code>false</code> (default value) when creating the resource. After creation can be updated and set to <code>true</code> to trigger a management subsystem rollback.
<code>\$IMAGE_REGISTRY</code>	Provide valid image registry to access the upgrade and version 10 artifacts. Must be provided if <code>DEFAULT_IMAGE_REGISTRY</code> is not set.
<code>\$IMAGE_PULL_SECRET S</code>	Valid image pull secrets to pull images from image registry specified in <code>\$IMAGE_REGISTRY</code> . Must be provided if <code>DEFAULT_IMAGE_PULL_SECRET</code> is not set.
<code>\$PROFILE</code>	Specifies the deployment profile, see Planning your deployment topology .
<code>\$STORAGE_CLASS NAM E</code>	Storage class name must be provided for all Volume Claim templates. The <code>default-class-name</code> will be used only if cluster's default-class-name has been set.
<code>\$VOLUME_SIZE</code>	<ul style="list-style-type: none"> <code>upgradeVolumeClaimTemplate</code>: The size of the PVC created for the extract job. Minimum size 50Gi. <code>databaseVolumeClaimTemplate</code>: - Minimum size 120 Gi <code>dbBackupVolumeClaimTemplate</code>: - Minimum size 120 Gi <code>dbArchiveVolumeClaimTemplate</code>: - Minimum size 30 Gi. <code>messageQueueVolumeClaimTemplate</code>: - Minimum size of 10 Gi

3. Edit the `license`: setting:

- Set `accept`: to `true` to accept the license. Note that the default value is `false`. If you do not accept the license, the Operator will not install the subsystem.
- Set `metric`: to track your product usage. Enter the unit of measure that is used for your program license:
 - `PROCESSOR_VALUE_UNIT` - Default value. If you leave the field blank, this value is used.
 - `MONTHLY_API_CALL` - Applies only to the IBM API Connect Hybrid Entitlement program. For information on tracking monthly call volume, see [Tracking API volume for auditing and compliance](#).
- Set `use`: to either `production` or `nonproduction`, to match the license you purchased.
- Set `license`: to the License ID for the version of API Connect that you purchased. See [API Connect licenses](#).

Example entry to accept the license for a production system:

```
license:
  accept: true
  metric: PROCESSOR_VALUE_UNIT
  use: production
  license: L-RJON-CEBLEH
```

4. Optional: - Database backup configuration can be specified in the upgrade CR.

You can also specify the database backup config after the management subsystem has been upgraded to version 10. If you choose to specify the database backup config in the management upgrade CR, the backup configuration will be carried forward to the v10 management subsystem. The upgraded management subsystem (version 10) will use this database backup config to perform backups as per the schedule.

Note:

You must:

- Use a new location (path) for the version 10 management database backup configuration; do not use the 2018 management backup path.
- Ensure that `Spec.ManagementSpec.DatabaseBackup.RestartDB` is set to `false`.
- Create a management backup secret before configuring database backups.

To review database backup configuration, including creation of a management backup secret, see [Configuring backup settings for fresh install of the Management subsystem](#).

Sample upgrade CR with database backup configuration:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementUpgradeFromV2018
metadata:
  name: mgmt
spec:
  subsystemName: mgmt
  triggerRollback: false
  upgradeVolumeClaimTemplate:
    storageClassName: local-storage
    volumeSize: 50Gi
  imageRegistry: apic-dev-docker-local.artifactory.swg-devops.com
  imagePullSecrets:
  - apic-registry-secret
  profile: nlxc4.m16
  license:
    accept: true
    metric: PROCESSOR_VALUE_UNIT
    use: production
    license: L-RJON-CEBLEH
  certManagerIssuer:
    name: selfsigning-issuer
    kind: Issuer
  managementSpec:
    databaseVolumeClaimTemplate:
      storageClassName: local-storage
      volumeSize: 120Gi
    dbBackupVolumeClaimTemplate:
      storageClassName: local-storage
      volumeSize: 120Gi
    dbArchiveVolumeClaimTemplate:
      storageClassName: local-storage
      volumeSize: 30Gi
```

```

messageQueueVolumeClaimTemplate:
  storageClassName: local-storage
  volumeSize: 10Gi
databaseBackup:
  credentials: ibmdbcreds
  host: s3.us-east.cloud-object-storage.appdomain.cloud/us-east
  path: $BUCKET_PATH/$BUCKET_NAME
  protocol: objstore
  restartDB:
    accept: false
  s3provider: ibm

```

5. Create the 2018 upgrade CR for management:

```
kubectl apply -f management.apiconnect.ibm.com_v1beta1_upgradefromv2018_cr.yaml -n <management-subsystem-namespace>
```

Note: `management-subsystem-namespace` in a single namespace APiConnect cluster will be same as the namespace as that of all the other subsystem namespaces.

The upgrade CR triggers the upgrade process for management.

6. Monitor upgrade progress:

a. Open a new terminal and tail upgrade-operator pod:

```
kubectl logs <upgrade-operator-pod> -n <management-subsystem-namespace>
```

b. Open a new terminal and check the status of the management upgrade:

```
kubectl get up2018m -n <management-subsystem-namespace>
```

Note: `up2018m` is the short name for management-upgrade-type and can be used to track the management upgrade progress. Each subsystem has a dedicated short name for upgrade:

Upgrade Subsystem	Short name
Management	up2018m
Analytics	up2018a
Portal	up2018p
Gateway	up2018g

Look for **READY** and **STATUS** in the resource's status to track your progress:

```

NAME          READY          STATUS          AGE
mgmt          14/14: Upgrade completed  Complete       12h

```

c. Once the upgrade process has completed the **extract** job, open a new terminal and monitor the progress of version 10 installation:

```
kubectl get apic -n <management-subsystem-namespace>
```

When the upgrade succeeds, the upgrade state displays **Upgrade is Complete**. Sample output:

```

NAME          STATUS          MESSAGE
AGE
managementdbupgrade.management.apiconnect.ibm.com/mgmt-up-4n6gq  Complete  Upgrade is Complete (DB Schema/data are up-to-date)  12h

NAME          READY          STATUS          AGE
managementupgradefromv2018.management.apiconnect.ibm.com/mgmt  14/14: Upgrade completed  Complete  12h

NAME          READY          STATUS          VERSION          RECONCILED VERSION          AGE
managementcluster.management.apiconnect.ibm.com/mgmt  16/16  Running  10.0.5.1  10.0.5.1-xxxx  12h

```

If you encounter an error, see [Rolling back a management upgrade to v2018](#).

7. Restart all nats-server pods by running the following command:

```
kubectl -n <namespace> delete po -l app.kubernetes.io/name=natscluster
```

8. If you did not specify backup settings in the upgrade CR in step 4, you must configure backup settings on the new version 10 management subsystem.

Note:

You must:

- Use a new location (path) for the version 10 management database backup configuration; do not use the 2018 management backup path.
- Ensure that `Spec.ManagementSpec.DatabaseBackup.RestartDB` is set to `false`.
- Create a management backup secret before configuring database backups.
- Expect that there will be a brief downtime upon configuring s3 backup settings in a new version 10 cluster.

To review database backup configuration, including creation of a management backup secret, see [Configuring backup settings for fresh install of the Management subsystem](#).

9. Save the upgrade logs after a successful upgrade. This is not mandatory but is recommended, to retain the ability to track the extract and load jobs.

a. Save the PVC logs after a successful upgrade.

The upgrade logs are stored in the `/upgrade` folder in the PVC. See [Logs for management upgrade](#)

b. Optionally, you can view logs of orphaned data from database upgrades. In the downloaded PVC contents, the logs are located in `logs/apim/orphans` and `logs/lur/orphans`. The data represents records which were deleted or updated because of invalid or missing references to data in other tables. If necessary, you can contact IBM Support to better understand the orphaned data.

c. Save the extract and load jobs logs. See [Logs for extract and load jobs](#)

10. We highly recommend that you retain the encryption secret and database credential secrets in a safe location, in case you need to restore during a disaster recovery scenario. Complete the steps in [Preparing the management subsystem for disaster recovery using S3 backups](#) or [Preparing the management subsystem for disaster recovery using SFTP backups](#).

Note: If you do not complete the steps in [Preparing the management subsystem for disaster recovery using S3 backups](#) or [Preparing the management subsystem for disaster recovery using SFTP backups](#), you will not be able to restore your management subsystem to a new cluster during a disaster recovery scenario.

11. Once the management subsystem has been upgraded, clear your browser cache and verify that you can log in to Cloud Manager UI and API manager UI and access the configuration menus.
12. Next, upgrade the Developer Portal subsystem. Continue with [Upgrading the Developer Portal subsystem from 2018 to v10](#).
Note: If the management subsystem upgraded encountered problems, see the instructions in the links at the end of this page. See also [Disaster recovery of a 2018 deployment](#)
 - [Rolling back a management upgrade to v2018](#)
If the upgrade of the management subsystem from v2018 to v10 fails, you can rollback to v2018.
 - [Cleaning up a failed management upgrade](#)
If the management upgrade fails, remove any unneeded artifacts.
 - [Stopping the v2018 to v10 management upgrade](#)
You can stop the v2018 to v10 upgrade process.
 - [Troubleshooting the management subsystem upgrade from v2018 to v10](#)
You can troubleshoot the upgrade of the management subsystem from v2018 to v10.

Rolling back a management upgrade to v2018

If the upgrade of the management subsystem from v2018 to v10 fails, you can rollback to v2018.

About this task

The management subsystem upgrade can only be rolled back to v2018 before the upgrade CR reaches the **Status: cleanUpInProgress** state. The **cleanUpInProgress** state is set after the load job has successfully completed.

Complete the following steps to trigger a rollback and then clean up the cluster after a management rollback:

Procedure

1. Determine the status of the upgrade:

- a. To view status:

```
kubectl get up2018m <management-subsystem-name> -n <management-subsystem-namespace>
```

- b. To view details of the upgrade:

```
kubectl describe up2018m <management-subsystem-name> -n <management-subsystem-namespace>
```

Check for the condition with **Status: True** to know about the current state of the upgrade.

A status condition of **RollbackRequired** indicates that the upgrade is unable to proceed and should be rolled back.

If any errors are encountered, see [Troubleshooting the management subsystem upgrade from v2018 to v10](#).

2. If a rollback is needed ensure that the required must gather logs have been collected. See [Logs for management upgrade](#) and [Logs for extract and load jobs](#)

3. The Operator does not perform an automated roll back. To trigger a rollback, update the upgrade CR:

- a. Edit the CR:

```
kubectl edit up2018m <management-subsystem-name> -n <management-subsystem-namespace>
```

- b. Under **spec**, add:

```
triggerRollback: true
```

- c. Save and exit to trigger roll back of v2018 upgrade.

- When performing a rollback, the upgrade operator clean-ups any v10 resources and brings back up the v2018 installation, as follows:
 - There will be no V10 resources created if the upgrade process gets reverted before running **extract job**.
 - You must wait for upgrade operator to delete v10 resources and scale up v2018 resources if the upgrade process gets reverted after running of extract job.
- The rollback performs the following steps:
 - Delete v10 secrets (if they were created). Note, however, that you must manually delete some secrets, as follows:

- Run **kubectl get secret -n <management-subsystem-namespace>** to fetch the secrets.
Returned secrets:

- **mgmt-*<uuid>*-postgres**
- **mgmt-*<uuid>*-postgres-backrest-repo-config**
- **mgmt-*<uuid>*-postgres-pgbouncer**
- **mgmt-*<uuid>*-postgres-postgres-secret**
- **mgmt-*<uuid>*-postgres-postgres-secret-apicuser**
- **mgmt-*<uuid>*-postgres-postgres-secret-replicator**
- **mgmt-*<uuid>*-postgres-primaryuser-secret**
- **mgmt-*<uuid>*-postgres-testuser-secret**
- **mgmt-ca**
- **mgmt-client**
- **mgmt-db-client-apicuser**
- **mgmt-db-client-pgbouncer**
- **mgmt-db-client-postgres**
- **mgmt-db-client-replicator**
- **mgmt-natscluster-mgmt**
- **mgmt-server**

- `mgmt-v2018-to-v10-extract-token-xxxxx`
 - `mgmt-v2018-to-v10-load-token-xxxxx`
- Use the following command to delete the returned secrets:

```
kubectl delete secret <secretName> -n <management-subsystem-namespace>
```

- Restarts the Cassandra cluster. Upgrade operator waits for Cassandra to be **Healthy** before scaling up v2018 deployments
 - Scale up v2018 deployment.
- Once rollback on management upgrade completes, operator status is set to **Failed**.

Cleaning up a failed management upgrade

If the management upgrade fails, remove any unneeded artifacts.

About this task

You can remove artifacts left from the failed upgrade, and then either retry the upgrade or stop the upgrade process.

Procedure

1. Determine if you have any v10 ingresses left after successful rollback, and delete them:
 - a. Check for ingresses:

```
kubectl get ingress -n <management-subsystem-namespace>
```

If any of the following ingresses are found, delete them manually:

- `<management-subsystem-name>-admin`
- `<management-subsystem-name>-api-manager`
- `<management-subsystem-name>-consumer-api`
- `<management-subsystem-name>-platform-api`

Note: v10 ingresses are not deleted if the upgrade operator entered **Failed** status during any of these upgrade steps:

- Creating postgres cluster
- Running load job
- Bringing up v10 deployments

- b. To manually delete the v10 ingresses after a rollback:

```
kubectl delete ingress <ingress-name> -n <management-subsystem-namespace>
```

Note: Delete only the v10 ingresses (if any). If not sure, compare the ingresses saved before performing the management upgrade to v10.

2. Use the following instructions to manually delete the PVC used by the upgrade process:

Note: Why is this step necessary?

The PVC used by extract and load job has the same name as that of the management upgrade CR name. Rolling back the management upgrade process does not delete the PVC used by extract/load jobs. This is done to ensure that you can access to the extract and load jobs after a rollback or a successful upgrade.

- a. If applicable, gather extract and load job logs. See [Logs for extract and load jobs](#).
- b. Gather logs for the upgrade operator. See [Logs for management upgrade](#).
- c. Manually delete the PVC:

Important: Save the PVC data before deleting it.

- i. Delete extract jobs:

```
kubectl delete jobs v2018-to-v10-extract -n <management-subsystem-namespace>
```

- ii. Delete load jobs:

```
kubectl delete jobs v2018-to-v10-load -n <management-subsystem-namespace>
```

- iii. Get the PVC name. The PVC name is the same as the name of the management upgrade CR.

```
kubectl get pvc -n <management-subsystem-namespace>
```

- iv. Delete the PVC:

```
kubectl delete pvc <management-upgrade-CR-name> -n <management-subsystem-namespace>
```

3. If necessary, delete the postgres PVCs.

The postgres PVCs must be deleted if you triggered a rollback on or after the upgrade reconciles the v10 cluster with only a postgres database.

- a. List the postgres PVCs:

```
kubectl get pvc -n <management-subsystem-namespace>
```

The following PVCs are used by `postgres` and must be deleted manually:

- `mgmt-xxxxxxx-postgres`
- `mgmt-xxxxxxx-postgres-pgbr-repo`
- `mgmt-xxxxxxx-postgres-wal`

- b. Delete the PVCs:

```
kubectl delete pvc mgmt-xxxxxxx-postgres -n <management-subsystem-namespace>
kubectl delete pvc mgmt-xxxxxxx-postgres-pgbr-repo -n <management-subsystem-namespace>
kubectl delete pvc mgmt-xxxxxxx-postgres-wal -n <management-subsystem-namespace>
```

4. Decide whether to troubleshoot and retry the management upgrade to v10, or to stop the upgrade process.

- To troubleshoot and retry, continue with Step 5.
- To stop the upgrade process, go to [Stopping the v2018 to v10 management upgrade](#).

5. To re-run the management upgrade from v2018 to v10:

a. Delete management upgrade CR:

```
kubectl delete up2018m <management-subsystem-name> -n <management-subsystem-namespace>
```

b. Fix the errors in your v2018 installation. See [Common errors on management upgrades](#).

c. Re-create the management upgrade v10 CR to trigger v10 management upgrade. See [Preparing the 2018 deployment for upgrade](#).

Stopping the v2018 to v10 management upgrade

You can stop the v2018 to v10 upgrade process.

About this task

If you do not want to continue a management upgrade in case of a failure, you can remove the upgrade artifacts.

Procedure

1. Ensure that either the management upgrade operator is in either **Failed** state, or the upgrade process has not yet created

`managementupgradefromv2018s.management.apiconnect.ibm.com`.

2. If you started the upgrade process and are ready to abandon it, clean up the artifacts from the management upgrade. See [Cleaning up a failed management upgrade](#).

3. Delete the all the upgrade CRDs and management upgrade resources. From the location of the downloaded files, run:

a. Delete all the left over upgrade resources:

- If upgrading a v2018 single namespace:

```
kubectl delete -f ibm-apiconnect-upgrade-crds.yaml
```

- If upgrading a v2018 multi-namespace:

```
kubectl delete -f ibm-apiconnect-upgrade-distributed.yaml -n <management-subsystem-namespace>
```

b. Delete all the left over v10 resources:

- If upgrading a v2018 single namespace:

```
kubectl delete -f ibm-apiconnect.yaml -n <v10-operator-namespace>
```

- If upgrading a v2018 multi-namespace:

```
kubectl delete -f ibm-apiconnect-distributed.yaml -n <v10-operator-namespace>
```

4. Delete DataPower resources:

```
kubectl delete -f ibm-datapower.yaml -n <gateway-namespace>
```

5. Delete `cert manager` and `ingress-issuer` resources applied in each namespace:

```
kubectl delete -f <path-to-cert-manager>/cert-manager-1.7.1.yaml
```

```
kubectl delete -f <path-to-ingress-issuer>/ingress-issuer-v1-alpha1.yaml -n <namespace>
```

Troubleshooting the management subsystem upgrade from v2018 to v10

You can troubleshoot the upgrade of the management subsystem from v2018 to v10.

Contents:

- [Management upgrade states](#)
- [Common errors on management upgrades](#)
- [Logs for management upgrade](#)
- [Logs for extract and load jobs](#)

Management upgrade states

Upgrade operator uses the following states to log the progress with management subsystem upgrades. The operator can be, but not necessarily be, in more than one state at a given point in time during management upgrade.

To determine the state of the operator, run:

```
kubectl describe up2018m <management-subsystem-name> -n <management-subsystem-namespace>.
```

Table 1. Management upgrade states

State	Description
Running	The management upgrade CR starts with this state. The management upgrade CR reports its progress in Reason and Message fields of this condition.

State	Description
CleanupInProgress	<p>The management upgrade CR enters this state after successfully creating all the v10 microservices. The upgrade CR remains in this state until all v2018 management components have been deleted. To track the progress of the v10 management cluster deployment run:</p> <pre>kubectl get <management-subsystem-name> -n <management-subsystem-namespace></pre> <p>The next condition set from here is Complete, upon successful upgrade completion.</p>
RollbackRequired	<p>The management upgrade CR enters this state when it hits any error during the upgrade process. Until the error gets fixed or upon setting triggerRollback: true, the upgrade CR will remain in this state.</p> <p>While in this state, you can either probe into the error and fix it or trigger a rollback by setting spec.triggerRollback: true to continue with the upgrade process. If you choose to trigger a rollback, the subsystem will be rolled-back and the original v2018 management subsystem installation will be brought up.</p> <p>The next condition set from this state is Failed.</p>
Failed	This condition is true when the upgrade process has failed.
Complete	This management upgrade CR enters this state upon a successful upgrade of the management subsystem.

Common errors on management upgrades

- Unable to perform backup due to incorrect backup configuration
- Unable to perform backup due to improper access rights to the backup folder
- Extract job stuck or not terminating with an Exit Code = 0
- Load job stuck or not terminating with an Exit Code = 0
- Unable to create v10 Management CR

Troubleshooting actions:

- Trigger a rollback. See [Rolling back a management upgrade to v2018](#).
- Review [Management upgrade states](#).
- Gather logs. See [Logs for management upgrade](#) and [Logs for extract and load jobs](#).
- Update the v2018 backup configuration. See [Backing up the management database in a Kubernetes environment](#)
- Restart the v2018 to v10 upgrade. See [Preparing the 2018 deployment for upgrade](#).
- Resolve any inconsistencies with the database. See [Preparing the 2018 deployment for upgrade](#).

Logs for management upgrade

The following logs are *must gather* logs. This means they are needed in order to diagnosis problems with the upgrade.

You must collect the following logs prior to setting **triggerRollback: true** in the management upgrade CR. Be sure to redirect the output of each to a suitable file location:

- Upgrade operator logs:

```
kubectl logs <upgrade_operator_pod> -n <management-subsystem-namespace>
```
- API Connect operator logs:

```
kubectl logs <apiconnect_operator_pod> -n <management-subsystem-namespace>
```
- Management CR description:

```
kubectl get up2018m <management-subsystem-name> -n <management-subsystem-namespace> -o yaml
```

Logs for extract and load jobs

- Example **kubectl** commands for viewing logs of extract and (or) load jobs:
 - Example for extract pod :

```
kubectl logs <mgmt-v2018-to-v10-extract-xxxxx>
```

To get information on the **extract** pod:

```
kubectl describe pod <mgmt-v2018-to-v10-extract-xxxxx>
```
 - Example for load pod:

```
kubectl logs <mgmt-v2018-to-v10-load-xxxxx>
```

To get information on the **load** pod:

```
kubectl describe pod <mgmt-v2018-to-v10-load-xxxxx>
```
- The management upgrade logs the following output for extract and (or) load jobs:
 - Any data that expires after a fixed time, such as access tokens and refresh token.
 - Invitations for creating provider org, catalog, space and consumer org as they have expiration time set.
 - Member invitations created in an admin org, provider org, catalog or space as they have expiration time set.
 - Logs and data directory from the PVC. These are the contents of /upgrade directory. You must have a logging service setup prior to attempting upgrade. For example, <https://logdna.com/>.
 - The output of **apicops preupgrade -m fix --upgrade_to v10**
 - The output of **apicops preupgrade -m report --upgrade_to v10** which is run before and after the fix option is run.

Upgrading the Developer Portal subsystem from 2018 to v10

You can upgrade the Developer Portal subsystem from 2018.4.1.20 or later to the latest release of v10.

Before you begin

Review the following requirements:

- The 2018 Developer Portal subsystem must be configured with either SFTP or S3 backups.
- All `portal-www` and `portal-db` pods must be up and ready. If they are not, the upgrade will not proceed.
- The environment must have enough storage space to accommodate the v10 PVCs.

If applicable, take the following optional steps:

1. Optional:

The portal subsystem upgrade is based around the backup and restore of the portal. All **INSTALLED** sites at the time of triggering the upgrade will be backed up. You can confirm the list of sites that will be backed up and subsequently upgraded by running the following command against one of the 2018 Portal www pods:

```
kubectl exec -it <portal_www_pod_name> -c admin -n <namespace> -- /opt/ibm/bin/list_sites -p
```

2. Optional: Make a note of the site name of any essential sites that you want to ensure are upgraded first. To view the site names:

```
kubectl exec -it <portal_www_pod_name> -c admin -n <namespace> -- /opt/ibm/bin/list_sites -p
```

You can provide these site names when creating the portal upgrade CR (see step 2 of the upgrade procedure), to ensure they are queued for upgrade in the order provided, before any remaining sites found in the specified backup location.

Important:

- The length of time it takes for the Portal subsystem upgrade to complete is directly related to the number of sites that are present in the portal. The more sites, the longer it takes. It is recommended that any non essential sites (especially those without any customizations) be removed prior to starting the upgrade and recreated once the upgrade has completed.
- Ensure that the specified backup location does not contain any old backups of portal sites that have subsequently been deleted. The upgrade procedure will restore (and upgrade) the backup for all sites found in the backup location. The backup location should only contain backups of the sites present and configured in API Manager at the time the upgrade is to take place.

About this task

The Portal upgrade procedure is primarily based around backing up a 2018 Portal subsystem and restoring it onto a newly deployed version 10 Portal cluster.

The steps which the operator goes through to upgrade the portal subsystem are as follows:

1. Upgrade initialization
2. Take a backup of the 2018 portal subsystem
3. Freeze the 2018 portal-www and portal-db pods
4. Create the 10 portal cluster CR.
5. Create a v10 portal restore CR to restore the 2018 portal backup onto the 10 portal subsystem and upgrade each portal site
6. Remove the 2018 portal subsystem resources
7. Set the **Complete** status condition of the portal upgrade CR to **true**.

To upgrade the Portal from 2018 to 10, complete the following steps:

Procedure

1. Verify that all prerequisites have been met:

- The Portal upgrade CR template file `portal.apiconnect.ibm.com_v1beta1_upgradefromv2018_cr.yaml` has been obtained from the API Connect upgrade `helper_files.zip` in IBM® API Connect <version> Operator Upgrade Release Files for Containers. See [Installing the files for upgrade from 2018](#).
- The apiconnect operator and apiconnect upgrade operator are installed and running: [Upgrading from 2018 on native Kubernetes](#)
- The Management subsystem upgrade is complete: [Upgrading the Management subsystem from 2018 to v10](#)
- All of the Portal prerequisites have been met: [Before you begin](#)

2. Populate the template upgrade CR for Portal `portal.apiconnect.ibm.com_v1beta1_upgradefromv2018_cr.yaml` with values for your deployment:

Table 1. Variables to set in the Upgrade CR template

Variable	Description
<code>\$NAME</code>	Name to uniquely identify this resource. Example: <code>pt1</code>
<code>\$PORTAL_SUBSYSTEM_NAME</code>	Must match the v2018's portal subsystem name. Use <code>apicup subsys list</code> to find the name. For example: <pre>\$ apicup subsys list Name Type Target mgmt ManagementSubsystem kubernetes gww5 GatewaySubsystem kubernetes gww6 GatewaySubsystem kubernetes portal PortalSubsystem kubernetes analytics AnalyticsSubsystem kubernetes</pre> Note: The value of the <code>subsystemName</code> property must match the name of your v2018 portal subsystem. You can also specify the optional <code>siteName</code> property in the <code>portalSpec</code> section to define the site name given to the v10 cluster.
<code>\$TRIGGER_ROLLBACK</code>	Can be <code>true</code> or <code>false</code> . Must initially be set to <code>false</code> (default value) when creating the resource. After creation can be updated and set to <code>true</code> to trigger a portal subsystem rollback.

Variable	Description
\$SITEURL1 - \$SITEURL2 - \$SITEURL3	PriorityList: entries prioritize the restore of any sites listed in the order in which they are provided. They will be queued for restore first, followed by any remaining sites present in the backup location. This is an optional property that can be omitted if the order of site restore or upgrades is not important. To obtain the site names, see Before you begin
\$IMAGE_REGISTRY	Provide valid image registry to access the upgrade and version 10 artifacts. Must be provided if <code>DEFAULT_IMAGE_REGISTRY</code> is not set.
\$IMAGE_PULL_SECRETS	Valid image pull secrets to pull images from image registry specified in <code>\$IMAGE_REGISTRY</code> . Must be provided if <code>DEFAULT_IMAGE_PULL_SECRET</code> is not set.
\$PROFILE	Specifies the deployment profile, see Planning your deployment topology .
\$STORAGE_CLASS_NAME	Storage class name must be provided for all Volume templates. The <code>default-class-name</code> will be used only if cluster's default-class-name has been set.
\$VOLUME_SIZE	<ul style="list-style-type: none"> <code>databaseVolumeClaimTemplate</code>: - Minimum size of 15Gi for Dev Profile and 30Gi for Prod. Adjust the size of the PVC according to the number of portal sites required <code>databaseLogsVolumeClaimTemplate</code>: - Minimum size of 6Gi for Dev Profile and 12Gi for Prod. <code>webVolumeClaimTemplate</code>: - Minimum size of 8Gi for Dev Profile and 12Gi for Prod. Adjust the size of the PVC according to the number of portal sites required. <code>backupVolumeClaimTemplate</code>: - Minimum size of 15Gi for Dev Profile and 30Gi for Prod. Adjust the size of the PVC according to the number of portal sites required. <code>adminVolumeClaimTemplate</code>: - Minimum size of 6Gi for Dev Profile and 12Gi for Prod. <code>certVolumeClaimTemplate</code>: - 4Gi is the recommended size for this template. Increasing this size does not provide any benefit.

Note: Note: The PVC sizes of your existing 2018 deployment should be specified in the first instance.

The PVC sizes allocated may need to be adjusted further (beyond the minimum and existing 2018 values) depending on the number and content of the sites required.

The minimum PVC sizes are large enough to cater for a small number of portal sites. As a rough estimate, for every 10 additional small portal sites (with a limited number of apps, apis, consumer orgs, members and portal content) the amount of space allocated should be increased in line with the following estimates:

- `databaseVolumeClaimTemplate`: +10 Gi
- `webVolumeClaimTemplate`: +1 Gi
- `backupVolumeClaimTemplate`: +2 Gi

If you have sites that have a lot of provider org content and lots of documentation, images, and additional content stored in the portal, then you may require a lot more space to be allocated to your PVCs. The following 2 PVCs do not need to be scaled based on the number of sites:

- `databaseLogsVolumeClaimTemplate`
- `adminVolumeClaimTemplate`

3. Edit the `license` setting:

- Set `accept`: to `true` to accept the license. Note that the default value is `false`. If you do not accept the license, the Operator will not install the subsystem.
- Set `use`: to either `production` or `nonproduction`, to match the license you purchased.
- Set `license`: to the License ID for the version of API Connect that you purchased. See [API Connect licenses](#).

Example entry to accept the license for a production system:

```
license:
  accept: true
  use: production
  license: L-RJON-CEBLEH
```

Example CR:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalUpgradeFromV2018
metadata:
  name: ptl
spec:
  subsystemName: ptl
  triggerRollback: false
```

```
imageRegistry: apic-dev-docker-local.artifactory.swg-devops.com
imagePullSecrets:
  - apic-registry-secret
profile: n3xc4.m8
license:
  accept: true
  use: production
  license: 'L-RJON-CEBLEH' # Refer license document https://ibm.biz/apiclicenses
certManagerIssuer:
  name: selfsigning-issuer
  kind: Issuer
```

```
portalSpec:
  databaseVolumeClaimTemplate:
    storageClassName: local-storage
    volumeSize: 30Gi
  databaseLogsVolumeClaimTemplate:
    storageClassName: local-storage
    volumeSize: 12Gi
  webVolumeClaimTemplate:
    storageClassName: local-storage
    volumeSize: 12Gi
  backupVolumeClaimTemplate:
    storageClassName: local-storage
    volumeSize: 30Gi
  adminVolumeClaimTemplate:
    storageClassName: local-storage
    volumeSize: 12Gi
  certVolumeClaimTemplate:
    storageClassName: local-storage
    volumeSize: 4Gi
```

4. Apply the portal upgrade from the 2018 CR:

```
kubectl apply -f portal.apiconnect.ibm.com_v1beta1_upgradefromv2018_cr.yaml -n <namespace>
```

5. Monitor the progression of the upgrade CR:

- Check the current state of the upgrade CR itself:

```
kubectl get up2018p -n <namespace>
```

- Monitor the activity of the upgrade by tailing the apiconnect-upgrade-operator pod logs:

```
kubectl logs -f <apiconnect-upgrade-operator-pod-name>
```

Upon completion the upgrade CR will display the status condition **Complete**.

6. Verify the state of your portal sites once the portal upgradefromv2018 has completed:

Run the `list_sites` command against one of the version 10 www pod admin containers

```
kubectl exec -it <v10_portal_www_pod_name> -c admin -n <namespace> -- /opt/ibm/bin/list_sites -p
```

You should see all sites installed. If you have a large number of sites then some may show as out of sync. Note that it can take a short while for API Manager to serve the snapshot requests for all sites, and for the portal to process them all.

Note: With the upgrade to version 10, the developer portal has had a major update to its look and appearance. This might require you to make some additional updates to pre-existing themes. Changes include:

- Forms such as Sign up and login are updated.
- The subscription wizard is refreshed.
- New styling for hovering on navigation bars.
- New concepts such as ghost and tertiary buttons.

If any sites have not successfully been installed then check the portal-www logs for more information. See:

- [Rolling back Developer Portal to v2018](#)
- [Troubleshooting Portal subsystem upgrade from v2018 to v10](#)

7. Retain the Portal subsystem encryption secret in a safe location, in case you need to restore the Portal subsystem during a disaster recovery scenario.

The encryption secret is created when the Portal subsystem is initially deployed. You can obtain the name of the encryption secret from the Portal subsystem custom resource once the installation is complete. For example:

a. Obtain the name of the encryptionSecret:

```
kubectl get ptl portal -o yaml | grep encryptionSecret
encryptionSecret: portal-enc-key
```

The name of encryption secret is `portal-enc-key`.

b. Save the secret yaml in a safe location:

```
kubectl get secret portal-enc-key -o yaml > portal-enc-key.yaml
```

8. Next upgrade the Analytics subsystem. Continue with [Upgrading the Analytics subsystem from v2018 to v10](#).

If your deployment does not use Analytics, continue with [Upgrading the gateway subsystem from 2018 to v10](#).

- [Rolling back Developer Portal to v2018](#)

If problems are encountered during the upgrade of Developer Portal to v10, you can rollback to v2018.

- [Troubleshooting Portal subsystem upgrade from v2018 to v10](#)

You can troubleshoot the Portal subsystem upgrade from v2018 to v10.

Rolling back Developer Portal to v2018

If problems are encountered during the upgrade of Developer Portal to v10, you can rollback to v2018.

About this task

The rollback process removes any v10 portal related resources and reinstates the v2018 resources. During rollback:

- The v2018 ingress resources are recreated.
- The v2018 portal `www` and `db` pods are unfrozen.
- Once the pods have come back up and reached **READY** status, any remaining portal upgrade related resources are removed.

If a portal upgrade CR has been rolled back, then in order to retry the upgrade a new portal upgrade CR must be created. It is recommended to delete the failed upgrade CR instance first.

The upgrade CR property `triggerRollback` provides the ability to define when a rollback should or should not be initiated. The `triggerRollback` property can take the following values:

Value	Description
<code>false</code>	Rollback of the portal subsystem will not be triggered. Default value
<code>true</code>	The upgrade of the portal subsystem will be rolled back immediately unless the upgrade process has already started to remove the v2018 portal resources, as indicated by the status condition CleanupInProgress is true. The upgrade CR is rejected if <code>triggerRollback</code> is set to <code>true</code> on creation.

You should gather log information before editing the upgrade CR and setting the `triggerRollback` value to `true`, as follows:

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place. If you are using a top-level CR you must edit the `multiSiteHA` section for the subsystem in the top-level CR, instead of directly in the subsystem CRs. If the subsystem

section is not included in the top-level CR, copy and paste the section from the subsystem CR to the top-level CR.

Procedure

1. Prior to setting `triggerRollback: true` in the upgrade CR, collect the following logs and details, and redirect the output of each to a suitable file location:

- a. Upgrade operator logs:

```
kubect1 logs <upgrade_operator_pod> -n <namespace>
```

- b. Apiconnect operator logs:

```
kubect1 logs <apiconnect_operator_pod> -n <namespace>
```

- c. Portal upgradefromv2018 CR description:

```
kubect1 describe up2018p <name> -n <namespace>
```

- d. Portal restore CR description (if present):

```
kubect1 describe portalrestore <subsystem_name> -n <namespace>
```

- e. Portal v10 subsystem CR description (if present):

```
kubect1 describe ptl <subsystem_name> -n <namespace>
```

- f. Portal v10 subsystem pod logs (if present):

- `Kubect1 logs <v10_portal_www> -c [admin/web] -n <namespace>`
- `Kubect1 logs <v10_portal_db> -c [db/dbproxy] -n <namespace>`
- `Kubect1 logs <v10_portal_nginx> -n <namespace>`

2. Specify the rollback in the upgrade CR:

- a. Edit the upgrade CR:

```
kubect1 edit up2018p <name> -n <namespace>
```

- b. Set `triggerRollback:`

```
triggerRollback: true
```

3. Monitor the progress of the rollback operation:

- Check the current status of the upgrade CR:

```
kubect1 describe up2018p <name> -n <namespace>
```

- Monitor the activity of the rollback operation by tailing the apiconnect-upgrade-operator pod logs:

```
kubect1 logs -f <apiconnect-upgrade-operator-pod-name>
```

Upon completion the upgrade CR will display the status condition `Failed`.

4. Once the rollback has completed you can delete the upgrade CR:

```
kubect1 delete up2018p <name> -n <namespace>
```

Troubleshooting Portal subsystem upgrade from v2018 to v10

You can troubleshoot the Portal subsystem upgrade from v2018 to v10.

Restored site failed to be upgraded successfully

If one or more of the restored sites failed to be upgraded successfully then check the portal-www admin container logs to see what prevented the site upgrade from completing successfully.

To trigger the site upgrade again, exec into the portal-www pod admin container and run the command:

```
upgrade_devportal -s <site_uuid> -p <platform>
```

- Site uuid can be obtained from the command `list_sites`
- Platform can be obtained from the command `list_platforms`

Upgrading the Analytics subsystem from v2018 to v10

It is not possible to directly upgrade the analytics subsystem from v2018 to v10.0.5. New v10.0.5 analytics subsystems must be deployed to replace your v2018 analytics subsystems.

- If you want to preserve your v2018 analytics data, export it to a backup location as explained in [Preparing the 2018 deployment for upgrade](#).
- Disassociate and delete your v2018 analytics subsystems before you upgrade your management subsystem, as explained in [Preparing the 2018 deployment for upgrade](#).

After your management subsystem upgrade is complete, deploy your v10.0.5 analytics subsystems: [Installing the Analytics subsystem](#).

Register your v10.0.5 analytics subsystems with your management subsystem: [Register an analytics service](#).

Upgrading the gateway subsystem from 2018 to v10

You can upgrade the DataPower Gateway subsystem from API Connect 2018.4.1.20 or later to the latest release of v10.

Before you begin

The apiconnect operator, apiconnect upgrade operator, and ibm-datapower operator must be installed and running, and the Management subsystem upgrade must be complete.

Note: Do not use this process if your 2018 deployment used a non-Kubernetes DataPower appliance or virtual appliance. This process is for 2018 deployments that configured gateway services through an API Connect gateway subsystem distribution on Kubernetes. For non-Kubernetes DataPower appliance or virtual appliance, see [Upgrading DataPower Gateway Service](#).

About this task

- How the gateway upgrade process works:
 1. Dual backend phase
 - The first part of the process uses a dual backend. The gateway upgrade starts by checking the replica count of the 2018 Gateway to set it as target for the upgraded version 10 gateway. The 2018 StatefulSet is scaled down by one, and the v10 Gateway is set to one replica.

While the first version 10 replica is starting, the api request traffic as well as admin traffic is only routed to 2018. Once the first v10 gateway pod is ready, admin traffic is routed to the version 10 gateway, and a `syncDelaySeconds` timer is used to allow the first version 10 gateway to synchronize with the API Connect Management subsystem.

After this delay, api traffic is routed to both 2018 pods and the first version 10 pod. The 2018 StatefulSet is then progressively scaled down, while the version 10 StatefulSet is scaled up to reach the target replicas matching the initial 2018 subsystem. Once the 2018 replicas is 0, the api traffic is only routed to version 10 Gateway (admin remains routed only to version 10 as well). The upgrade CR transitions then to "single backend".

While this dual backend upgrade is happening, it is possible to update the upgrade CR to set a value `triggerRollback: true` which will remove the version 10 StatefulSet and scale back the 2018 StatefulSet to its initial replica count. The api and admin traffic is then also only routed to 2018 pods.
 2. Single backend phase

Once the target replica for version 10 is reached, the 2018 artifacts are deleted, and the upgrade CR transitions then to "complete". During this single backend upgrade phase, it is no longer possible to trigger a rollback.
 3. Completion

Once the upgrade has completed successfully the state of the upgrade CR is `Complete`.
- When the Gateway is upgraded from 2018 the following information is derived directly from the 2018 deployment:
 - The `apic-gw-service-ingress` certificate, per 2018 certificate reference, is carried over as the `-apic-gateway-service-tls` secret and pointed to by the version 10 `.cr.Spec.APICGatewayServiceTLS`
 - The gateway-peering certificate, per 2018 certificate reference, is carried over as the `-apic-gateway-peering-tls` secret and pointed to by `.cr.Spec.APICGatewayPeeringTLS`.
 - The admin user secret from 2018 is carried over in version 10 as a `-datapower-admin-credentials` secret pointed to by `.cr.Spec.AdminUser`.
 - The v5 compatibility mode setting of the 2018 deployment is carried over as the version 10 `.cr.Spec.APICGatewayServiceV5CompatibilityMode`.
 - Whether TMS is enabled in 2018 is carried over as the version 10 `.cr.Spec.TokenManagementService.Enabled` flag.
 - The 2018 deployment resource limits (for cpu and memory) for the statefulset pods are carried over as the resource requests and limits (for cpu and memory) for the version 10 deployment.
- Additional Gateway configuration from 2018:
 - In case TMS is used, the storage class and size from 2018 is not carried over; it must instead be specified in the upgrade CR as `.upgradeCR.Spec.TokenManagementServiceVolumeClaimTemplate`. If size is not specified, the default size is 30Gi.
 - If not specified, the default storage class is set to the default storage class defined in the cluster. If there is no default storage class on the cluster, an error will occur: `Cannot find default storage class`.
 - In 2018, you used an extra values file when a specialized DataPower configuration was required. In version 10, operators do not use extra values files. Instead, you can set the following properties in the CR:
 - `AdditionalDomainConfig` - see :
 - [Domains](#)
 - [Guides: Domain Configuration](#)
See the guide for more detail on how to set up the structure, and components such as `Secret`, `ConfigMap`, and more.
 - [Features: Configuration Management](#)
 - `AdditionalExtraExe` - see [Extraexe](#)
 - `AdditionalInitCmds` - see [Initcmds](#)
- Upgrade considerations:
 - When upgrading from a Dev mode 2018 gateway deployment (single replica, rather than the three replicas in Standard mode), there will be downtime during the upgrade. In this case, the 2018 gateway StatefulSet is scaled down before the v10 StatefulSet is scaled up. API requests to the gateway cannot succeed until the version 10 upgrade is completed, or until the version 10 upgrade is rolled back to 2018.
 - During the upgrade, rate-limiting of API transactions are not properly enforced. In this case, the 2018 gateway cluster has no way to share the rate-limit information with the version 10 gateway cluster.

Procedure

1. Verify that all prerequisites are met. See [Before you begin](#).
2. Populate the template upgrade CR for Gateway `gateway.apiconnect.ibm.com_v1beta1_upgradefromv2018_cr.yaml` with values for your deployment:

Table 1. Gateway service upgrade CR variables

Variable	Description
<code>\$NAME</code>	Must be unique for each Gateway service upgraded
<code>\$GATEWAY_SUBSYSTEM_NAME</code>	<p>Must match the name of the Gateway subsystem to be upgraded as it is named in the v2018 API Connect <code>apicup</code> installer project. Use <code>apicup subsys list</code> to display names of Gateway subsystems in the v2018 API Connect <code>apicup</code> installer project. For example:</p> <pre>\$ apicup subsys list Name Type Target gmt ManagementSubsystem kubernetes gwv5 GatewaySubsystem kubernetes gwv6 GatewaySubsystem kubernetes portal PortalSubsystem kubernetes analytics AnalyticsSubsystem kubernetes</pre> <p>In this example, the gateway upgrade CR should specify <code>subsystemName: gwv5</code> to upgrade the <code>gwv5</code> gateway, or <code>subsystemName: gwv6</code> to upgrade the <code>gwv6</code> gateway. Note that if the 2018 deployment contained multiple gateway services, the gateway services must be updated individually, by creating one upgrade CR per gateway service to be upgraded.</p>
<code>\$TRIGGER_ROLLBACK</code>	Can be <code>true</code> or <code>false</code> . Must initially be set to <code>false</code> (default value) when creating the resource. After creation can be updated and set to <code>true</code> to trigger a portal subsystem rollback.
<code>\$IMAGE_REGISTRY</code>	Provide valid image registry to access the upgrade and version 10 artifacts. Must be provided if <code>DEFAULT_IMAGE_REGISTRY</code> is not set.
<code>\$IMAGE_PULL_SECRETS</code>	Valid image pull secrets to pull images from image registry specified in <code>\$IMAGE_REGISTRY</code> . Must be provided if <code>DEFAULT_IMAGE_PULL_SECRET</code> is not set in the upgrade operator deployment yaml file (<code>ibm-apiconnect-upgrade.yaml</code>)
<code>\$PROFILE</code>	Specifies deployment profile of the version 10 installation to upgrade to. See Planning your deployment topology for details of the available profiles.
<code>\$SYNC_DELAY_SECONDS</code>	The delay is the amount of time given to the first version 10 DataPower pod to complete its synchronization with the API Connect Management subsystem. After that delay, api request traffic starts being forwarded to the first version 10 pod. The version 10 StatefulSet is then scaled up progressively to the target replicas, allowing about 2 minutes for each new pod to be synchronized with the others. The default is 600 seconds (10 minutes)
<code>\$STORAGE_CLASS_NAME</code>	If the Gateway to be upgraded has Token Management Service (TMS) enabled, the <code>tokenManagementServiceVolumeClaimTemplate</code> settings must be populated. Provide a storage class name. The default-class-name will be used only if cluster's default-class-name has been set. For example, <pre>storageClassName: local-storage</pre>
<code>\$VOLUME_SIZE</code>	If the Gateway to be upgraded has Token Management Service (TMS) enabled, the <code>tokenManagementServiceVolumeClaimTemplate</code> settings must be populated. Provide a volume size. The minimum size is 30Gi.

3. Edit the `license` setting:
 - a. Set `accept`: to `true` to accept the license. Note that the default value is `false`. If you do not accept the license, the Operator will not install the subsystem.
 - b. Set `use`: to either `production` or `nonproduction`, to match the license you purchased.
 - c. Set `license`: to the License ID for the version of API Connect that you purchased. See [API Connect licenses](#).

Example entry to accept the license for a production system:

```
license:
  accept: true
  use: production
  license: L-RJON-CEBLEH
```

Example CR:

```
apiVersion: gateway.apiconnect.ibm.com/v1beta1
kind: GatewayUpgradeFromV2018
metadata:
  name: $NAME # name to uniquely identify this resource. Example: `gwupgrade`
spec:
  subsystemName: $GATEWAY_SUBSYSTEM_NAME # must be same as v2018's gateway subsystem name. Find out using the following
v2018 installer command: apicup subsys list
  triggerRollback: $TRIGGER_ROLLBACK # takes true or false. Must initially be set to false (default value) when creating
the resource. After creation can be updated and set to true to trigger a gateway subsystem rollback
  imageRegistry: $IMAGE_REGISTRY # provide valid image registry to access the upgrade and v10 artifacts. Must be provided
if DEFAULT_IMAGE_REGISTRY is not set.
  imagePullSecrets:
    - $IMAGE_PULL_SECRETS # valid image pull secrets to pull images from image registry specified above. Must be provided
if DEFAULT_IMAGE_PULL_SECRET is not set.
  profile: $PROFILE # specifies deployment profile of the v10 gateway installation to upgrade to.
  license: # must be defined
    accept: true
    use: production
    license: L-RJON-CEBLEH # Refer license document https://ibm.biz/apiclicenses
  gatewaySpec:
    syncDelaySeconds: $SYNC_DELAY_SECONDS # amount of time given to the first v10 Datapower pod to complete its
synchronization with the API Connect Management subsystem. Defaults to 600 seconds if not defined.
    tokenManagementServiceVolumeClaimTemplate: # should be populated if the Gateway to be upgraded has TMS enabled. Refer
upgrade doc for more info.
      storageClassName: $STORAGE_CLASS_NAME # storage class name must be provided. default-class-name will be used only if
cluster's default-class-name has been set.
      volumeSize: $VOLUME_SIZE # must be atleast 30Gi
```

4. Apply the gateway `upgradefromv2018` CR with the following command:

```
kubectl apply -f gateway.apiconnect.ibm.com_v1beta1_upgradefromv2018_cr.yaml -n <namespace>
```

5. Monitor the progression of the upgrade CR:

- To check the current state of the upgrade CR:

```
kubect1 get up2018g -n <namespace>
```

- To monitor the activity of the upgrade, tail the apiconnect-upgrade-operator pod logs:

```
kubect1 logs -f <apiconnect-upgrade-operator-pod-name> -n <namespace>
```

Once completed the upgrade CR will enter the status condition **Complete**.

6. Continue with [Post-upgrade steps for upgrading to v10](#).

- [Rollback of gateway subsystem to v2018](#)

If the upgrade from v2018 to v10 does not succeed, you can rollback the gateway subsystem to v2018.

Rollback of gateway subsystem to v2018

If the upgrade from v2018 to v10 does not succeed, you can rollback the gateway subsystem to v2018.

About this task

The upgrade CR has an optional property `triggerRollback` which provides the ability to define when a rollback should or should not be initiated. The `triggerRollback` property can take the following values:

Value	Description
<code>false</code>	Rollback of the gateway subsystem will not be triggered. Default value
<code>true</code>	The upgrade of the gateway subsystem will be rolled back immediately unless the upgrade process has already moved to a status condition of Complete . The upgrade CR is rejected if <code>triggerRollback</code> is set to <code>true</code> on creation.

You can rollback the gateway subsystem upgrade during the upgrade's dual backend phase, but not during the single backend phase:

1. Dual backend phase

- The first part of the process uses a dual backend. The gateway upgrade starts by checking the replica count of the v2018 Gateway to set it as target for the upgraded v10 gateway. The v2018 StatefulSet is scaled down by one, and the v10 Gateway is set to one replica.

While the first v10 replica is starting, the api request traffic as well as admin traffic is only routed to v2018. Once the first v10 gateway pod is ready, admin traffic is routed to the v10 gateway, and a `syncDelaySeconds` timer is used to allow the first v10 gateway to synchronize with the API Connect Management subsystem.

After this delay, api traffic is routed to both v2018 pods and the first v10 pod. v2018 StatefulSet is then progressively scaled down, while the v10 StatefulSet is scaled up to reach the target replicas matching the initial v2018 subsystem. Once the v2018 replicas is 0, the api traffic is only routed to v10 Gateway (admin remains routed only to v10 as well). The upgrade CR transitions then to "single backend".

While this dual backend upgrade is happening, it is possible to update the upgrade CR to add a value `triggerRollback: true` which will remove the v10 StatefulSet and scale back the v2018 StatefulSet to its initial replica count. The api and admin traffic is then also only routed to v2018 pods.

2. Single backend phase

Once the target replica for v10 is reached, the v2018 artifacts are deleted, and the upgrade CR transitions then to **Complete**. During this single backend phase, you cannot trigger a rollback.

Procedure

1. Ensure that the state of upgrade CR has not reached the **Complete** status.
2. Specify the rollback in the upgrade CR:
 - a. Edit the upgrade CR:

```
kubect1 edit up2018g <upgrade_cr_name> -n <namespace>
```

- b. Change the `triggerRollback` value to `true`:

```
triggerRollback: true
```

When the upgrade was rolled back, then after the v2018 resources are scaled back to the initial replicas and the v10 resources are deleted, the upgrade CR is marked **RollbackComplete**.

Post-upgrade steps for upgrading to v10

After you upgrade API Connect from v2018 to v10, complete these final steps.

Procedure

1. Enable the Analytics subsystem as explained in [Enabling Analytics after upgrading](#).
2. To ensure that the v10 user interface works correctly, clear your browser cache.
3. Clear out v2018 resources:
 - a. List the helm charts.

```
helm list
```

b. Note the **Name** of the release relating to each of the following v2018 chart names:

- `apic-analytics-x.x.x`
- `apiconnect-x.x.x`
- `cassandra-operator-x.x.x`
- `apic-portal-x.x.x`
- `dynamic-gateway-service-x.x.x`

c. Clear out v2018 resources. For each **Release Name**, run:

```
helm delete <Release-Name> --purge
```

d. If helm version is 3.5.1, use the following command to dry run uninstalling of the helm release:

```
helm uninstall <Release-Name> -n <management_subsystem_namespace> --dry-run
```

Run the below command to uninstall the v2018 helm release

```
helm uninstall <Release-Name> -n <management_subsystem_namespace>
```

4. Manually delete any remaining v2018 pods for the current release:

```
kubectl delete pod <releaseName>-apim-schema-init-job-xxxx -n <ManagementNamespace>
```

and

```
kubectl delete pod <releaseName>-lurschema-init-job-xxxx -n <ManagementNamespace>
```

5. Delete secrets:

a. Search for cassop (cassandra) secrets with the below names:

- `cassandra-backup-auth-secret-xxx`,
- `cassop-client-certs-xxx`,
- `cassop-server-certs-xxx`

b. For each secret, run:

```
kubectl delete secret <Listed_Cassop_Secrets_from_Search> -n <managementNamespace>
```

c. Manually delete the following secrets:

- `kubectl delete secret analytics-proxy-velox-certs-xxx -n <managementNamespace>`
- `kubectl delete secret apim-elasticsearch-velox-certs-xxx -n <managementNamespace>`
- `kubectl delete secret apim-ui-basic-auth -n <managementNamespace>`
- `kubectl delete secret apim-velox-certs-xxx -n <managementNamespace>`
- `kubectl delete secret client-downloads-server-velox-certs-xxx -n <managementNamespace>`
- `kubectl delete secret db-client-velox-certs-xxx -n <managementNamespace>`
- `kubectl delete secret db-velox-certs-xxx -n <managementNamespace>`
- `kubectl delete secret juhu-velox-certs-xxx -n <managementNamespace>`
- `kubectl delete secret ldap-velox-certs-xxx -n <managementNamespace>`
- `kubectl delete secret lur-velox-certs-xxx -n <managementNamespace>`
- `kubectl delete secret migration-inputs-xxx -n <managementNamespace>`
- `kubectl delete secret migration-velox-certs-xxx -n <managementNamespace>`
- `kubectl delete secret plugin-certs-xxx -n <managementNamespace>`
- `kubectl delete secret ui-velox-certs-xxx -n <managementNamespace>`

Attention: When you upgrade to v10 from v2018, some external certificates are carried over from the v2018 deployment and are not managed by cert-manager. When you update any of the retained certificates, you must restart the affected pods manually. For a list of the certificates that are retained after an upgrade and must be updated manually, see [V2018 upgrade: List of certificates to update manually](#) in the [Renewing TLS certificates](#) section of this documentation.

6. Delete any remnant v2018 Analytics cronjob pods.

Two cronjob pods can remain after the upgrade with names matching the format shown below:

- `r*****-analytics-cronjobs-retention-*****-*****`
- `r*****-analytics-cronjobs-rollover-*****-*****`

These are remnant pods from v2018 and are now redundant. To remove them:

a. List the pods:

```
kubectl get pods -n <namespace>
```

b. Delete the pods:

```
kubectl delete pods <name-of-remnant-cronjob-pod> -n <namespace>
```

7. Optional: Configure additional features related to inter-subsystem communication security, such as CA verification and JWT security: [Optional post-upgrade steps for upgrade to 10.0.5.3 \(or later\) from earlier 10.0.5 release](#).

Disaster recovery of a 2018 deployment

You can optionally return to a 2018 deployment, if necessary, after completion of a successful upgrade from 2018 to v10.

About this task

You can return to the 2018 system by performing a disaster recovery restore on each subsystem.

Note:

- You must have upgraded the subsystems in the order specified in [Upgrade requirements and limitations](#).
- To return to a 2018 system, you must remove all version 10 subsystems. To uninstall API Connect entirely, see [Removing API Connect](#). You can optionally follow the links below to remove individual subsystems, but you must remove all subsystems to successfully restore a 2018 system.

Procedure

1. Management subsystem

Perform disaster recovery of the management subsystem if the 2018 Management subsystem was successfully upgraded and all 2018 Management resources were deleted but you now need to revert to version 2018. You also must perform disaster recovery of the management subsystem if the 2018 Portal subsystem was successfully upgraded and all 2018 Portal resources were deleted, but you now need to revert to version 2018.

- a. To uninstall the v10 management subsystem, see [Uninstalling API Connect subsystems](#).
 - To uninstall API Connect entirely, see [Removing API Connect](#).
- b. Follow the 2018 instructions in [Restoring the management database in a Kubernetes environment](#).

Note:

- You will need to restore both the Management subsystem and the Portal subsystem. Be sure to choose backups from the same time to perform a restore.
- You must perform disaster recovery for Portal, as described in Step 2, immediately after the Management restore. Carefully follow all the prerequisites of 2018 Portal disaster recovery.

2. Developer Portal subsystem

Perform disaster recovery of the 2018 Portal if the 2018 Portal subsystem was successfully upgraded and all 2018 portal resources were deleted but you now need to revert to version 2018

- a. You must first complete disaster recovery for the Management subsystem, as described in Step 1, and then perform Portal disaster recovery.
- b. Uninstall the version 10 Portal:
 - To uninstall just the Portal subsystem, see [Uninstalling API Connect subsystems](#).
 - To uninstall the entire API Connect deployment, see [Removing API Connect](#).
- c. Follow the instructions in [Backing up and restoring the Developer Portal in a Kubernetes environment](#), being sure to follow all the prerequisites. When restoring, be sure to choose backups for both the Management subsystem and the Portal subsystem from the same time.

3. Gateway subsystem

You will want to perform disaster recovery of the 2018 Gateway subsystem if the 2018 Gateway subsystem was successfully upgraded and all the 2018 gateway resources were deleted, but you now need to revert to 2018.

Note that although the upgrade from 2018 to 10.0.1.x-eus Gateway on Kubernetes is designed for zero downtime (when using a 3 node setup), reverting to the 2018 gateway will incur downtime.

You must perform Management disaster recovery first and then perform Gateway disaster recovery.

- a. Uninstall the v10 Gateway:
 - To uninstall just the Gateway subsystem, see [Uninstalling API Connect subsystems](#).
 - To uninstall the entire API Connect deployment, see [Removing API Connect](#).
- b. Install the 2018 Gateway subsystem. See [Installing the Gateway subsystem into a Kubernetes environment](#).

4. Analytics subsystem


You will want to perform disaster recovery of the 2018 Analytics subsystem if your 2018 Analytics subsystem was successfully upgraded and all the 2018 Analytics resources were deleted, but you now need to revert to 2018.

If Management and Portal subsystems are already upgraded to 10.0.1.x-eus, you must complete disaster recovery for Management subsystem, as described in Step 1, and then disaster recovery for the Portal subsystem, as described in Step 2.

- a. Uninstall the v10 Analytics subsystem:
 - To uninstall just the Analytics subsystem, see [Uninstalling API Connect subsystems](#).
 - To uninstall the entire API Connect deployment, see [Removing API Connect](#).
- b. Install 2018 Analytics. See [Installing the Analytics subsystem into a Kubernetes environment](#).
- c. Restore the 2018 backup that you took before upgrading. See [Restoring the Analytics database](#).
- d. Re-associate the Analytics service with your Gateways.

Troubleshooting upgrades on Kubernetes

Review the following troubleshooting guidance if you encounter a problem while upgrading API Connect on Kubernetes.

Note: In the  Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.

- [PgBouncer pod enters a CrashLoopBackoff state when upgrading to API Connect 10.0.5.3](#)
- [Upgrading a 3 node profile might result in some portal-db/www pods being stuck in the Pending state](#)

PgBouncer pod enters a CrashLoopBackoff state when upgrading to API Connect 10.0.5.3

This issue might be caused by a corrupt compliance entry in the pgbouncer.ini file. When the operator is upgraded to v10.0.5.3 but the ManagementCluster CR is not yet upgraded, the operator might update the pgbouncer.ini file in the PGBouncer secret with the older ManagementCluster CR's profile file, which does not contain any value for the `compliance pool_size`. As a result, the value gets incorrectly set to the string `<no value>`.

In the majority of cases this bad configuration update is temporary, until the ManagementCluster CR's version is also updated as part of the upgrade process and the problem is resolved. The issue will only become evident if the pgbouncer pod restarts prior to the ManagementCluster CR's version field being updated.

After the operator updates, if the other pods are restarted (specifically the pgbouncer pod) before the ManagementCluster CR's version is updated, then the pgbouncer pod will get stuck in a `CrashLoopBackOff` state due to the missing value for the `compliance pool_size` setting.

The following example shows what the error looks like in the pgbouncer logs:

```
Wed May 17 09:00:02 UTC 2023 INFO: Starting pgBouncer..
2023-05-17 09:00:02.897 UTC [24] ERROR syntax error in connection string
2023-05-17 09:00:02.897 UTC [24] ERROR invalid value "host=mgmt-a516d013-postgres port=5432 dbname=compliance pool_size=<no value>" for parameter compliance in configuration (/pgconf/pgbouncer.ini:7)
2023-05-17 09:00:02.897 UTC [24] FATAL cannot load config file
```

Resolve the issue by running the commands in the following script:

```
NAMESPACE=<mgmt-namespace>
BOUNCER_SECRET=<management-prefix>-postgres-pgbouncer-secret
TEMP_FILE=/tmp/pgbouncer.ini

**Step 1 - Common: Get the existing pgbouncer.ini file**
kubectl get secret -n $NAMESPACE $BOUNCER_SECRET -o jsonpath='{.data.pgbouncer\.ini}' | base64 -d > $TEMP_FILE

**Step 2 - Linux version: Update the file and use it to patch the Secret on the cluster**
sed 's/<no value>/20/' $TEMP_FILE | base64 -w0 | xargs -I{} kubectl patch secret -n $NAMESPACE $BOUNCER_SECRET --type='json' -p='[{"op": "replace", "path": "/data/pgbouncer.ini", "value": {}}]'
```

```
**Step 2 - Mac version: Update the file and use it to patch the Secret on the cluster**
sed 's/<no value>/20/' $TEMP_FILE | base64 -b0 | xargs -S2000 -I{} kubectl patch secret -n $NAMESPACE $BOUNCER_SECRET --type='json' -p='[{"op": "replace", "path": "/data/pgbouncer.ini", "value": {}}]'
```

```
# Step 3 - Common: Restart pgbouncer to pick up the updated Secret configuration
kubectl delete pod <bouncer_pod_name> -n $NAMESPACE
```

Upgrading a 3 node profile might result in some portal-db/www pods being stuck in the Pending state

IBM® API Connect 10.0.3.0 introduces the pod anti-affinity required rule, meaning that in a 3 node profile deployment, all 3 db and www pods can run only if there are at least 3 running worker nodes. This rule can cause some upgrades to version 10.0.3.0 or later to become stuck in the `Pending` state, in which case some extra steps are needed during the upgrade to workaroud the issue. See the following example for detailed information about the issue, and how to continue with the upgrade.

Important: You must have a backup of your current deployment before starting the upgrade.

The following example shows 3 worker nodes:

```
$ kubectl get nodes
NAME                STATUS    ROLES    AGE    VERSION
apimdev0103         Ready    worker   42m    v1.20.0
apimdev0129         Ready    worker   45m    v1.20.0
apimdev1066         Ready    worker   39m    v1.20.0
```

The pods have been scheduled across only 2 of the 3 worker nodes due to a transient problem with `apimdev1066`, as shown in the following pod list. Pods without persistent storage, such as `nginx-X`, can be rescheduled to `apimdev1066` as soon as they are restarted, but any pods with persistent local storage, such as `db-X` and `www-X`, have to be rescheduled onto the same worker node as that is where their files live.

```
$ kubectl get po -o wide
NAME                READY    STATUS    RESTARTS    AGE    IP                NODE                NOMINATED NODE
READINESS GATES
ejs-portal-nginx-84f57ffd8c-hbf66  1/1    Running    0            5m12s  888.16.109.208   apimdev0103        <none>        <none>
ejs-portal-nginx-84f57ffd8c-mvq96  1/1    Running    0            5m12s  888.16.142.215   apimdev0129        <none>        <none>
ejs-portal-nginx-84f57ffd8c-vpmtl  1/1    Running    0            5m12s  888.16.142.214   apimdev0129        <none>        <none>
ejs-portal-sitel-db-0              2/2    Running    0            4m39s  888.16.109.209   apimdev0103        <none>        <none>
ejs-portal-sitel-db-1              2/2    Running    0            6m37s  888.16.109.206   apimdev0103        <none>        <none>
ejs-portal-sitel-db-2              2/2    Running    0            4m39s  888.16.142.216   apimdev0129        <none>        <none>
ejs-portal-sitel-www-0             2/2    Running    0            4m9s   888.16.109.210   apimdev0103        <none>        <none>
ejs-portal-sitel-www-1             2/2    Running    0            6m37s  888.16.142.213   apimdev0129        <none>        <none>
ejs-portal-sitel-www-2             2/2    Running    0            4m9s   888.16.142.217   apimdev0129        <none>        <none>
ibm-apiconnect-75b47f9f87-p25dd    1/1    Running    0            5m12s  888.16.109.207   apimdev0103        <none>        <none>
```

The upgrade to version 10.0.3.0 is started and the following pod list is observed:

```
$ kubectl get po -o wide
NAME                READY    STATUS    RESTARTS    AGE    IP                NODE                NOMINATED NODE
READINESS GATES
ejs-portal-nginx-84f57ffd8c-hbf66  1/1    Running    0            10m    888.16.109.208   apimdev0103        <none>        <none>
ejs-portal-nginx-84f57ffd8c-mvq96  1/1    Running    0            10m    888.16.142.215   apimdev0129        <none>        <none>
ejs-portal-nginx-84f57ffd8c-vpmtl  1/1    Running    0            10m    888.16.142.214   apimdev0129        <none>        <none>
ejs-portal-sitel-db-0              2/2    Running    0            10m    888.16.109.209   apimdev0103        <none>        <none>
ejs-portal-sitel-db-1              0/2    Pending    0            91s    <none>           <none>           <none>
ejs-portal-sitel-db-2              2/2    Running    0            2m41s  888.16.142.218   apimdev0129        <none>        <none>
```

ejs-portal-sitel-www-0	2/2	Running	0	9m51s	888.16.109.210	apimdev0103	<none>	<none>
ejs-portal-sitel-www-1	2/2	Running	0	12m	888.16.142.213	apimdev0129	<none>	<none>
ejs-portal-sitel-www-2	0/2	Pending	0	111s	<none>	<none>	<none>	<none>
ibm-apiconnect-75b47f9f87-p25dd	1/1	Running	0	10m	888.16.109.207	apimdev0103	<none>	<none>

The pod list shows that **db-2** has restarted, and has been rescheduled to **apimdev0129** as there were no other db pods running on that node. However, **db-1** and **www-2** are both stuck in **Pending** state, as there is already a pod of the same type running on the worker node that is hosting the local storage that they are bound to. If you run a **describe** command of either pod you will see the following output:

```
Events:
  Type            Reason              Age             From              Message
  ----            -
  Warning         FailedScheduling   10s (x4 over 2m59s)  default-scheduler  0/3 nodes are available: 1 node(s) didn't match pod affinity/anti-affinity, 1 node(s) didn't match pod anti-affinity rules, 2 node(s) had volume node affinity conflict.
```

To resolve this situation you need to delete the PVCs for each pod, and then delete the pod itself, so that Kubernetes will regenerate the PVCs and schedule the pod on the worker node that does not have the anti-affinity conflict.

Therefore, for the **db-1** pod the following commands must be run:

```
$ kubectl get pvc | grep ejb-portal-sitel-db-1
db-ejs-portal-sitel-db-1      Bound      local-pv-fa445e30    250Gi      RWO          local-storage    15m
dblogs-ejs-portal-sitel-db-1 Bound      local-pv-d57910e7    250Gi      RWO          local-storage    15m

$ kubectl delete pvc db-ejs-portal-sitel-db-1 dblogs-ejs-portal-sitel-db-1
persistentvolumeclaim "db-ejs-portal-sitel-db-1" deleted
persistentvolumeclaim "dblogs-ejs-portal-sitel-db-1" deleted

$ kubectl delete po ejb-portal-sitel-db-1
pod "ejb-portal-sitel-db-1" deleted
```

For the **www-2** pod the following commands must be run:

```
$ kubectl get pvc | grep ejb-portal-sitel-www-2
admin-ejs-portal-sitel-www-2 Bound      local-pv-48799536    245Gi      RWO          local-storage    51m
backup-ejs-portal-sitel-www-2 Bound      local-pv-a93f5607    245Gi      RWO          local-storage    51m
web-ejs-portal-sitel-www-2   Bound      local-pv-facd4489    245Gi      RWO          local-storage    51m

$ kubectl delete pvc admin-ejs-portal-sitel-www-2 backup-ejs-portal-sitel-www-2 web-ejs-portal-sitel-www-2
persistentvolumeclaim "admin-ejs-portal-sitel-www-2" deleted
persistentvolumeclaim "backup-ejs-portal-sitel-www-2" deleted
persistentvolumeclaim "web-ejs-portal-sitel-www-2" deleted

$ kubectl delete po ejb-portal-sitel-www-2
pod "ejb-portal-sitel-www-2" deleted
```

If the PVC has **persistentVolumeReclaimPolicy: Delete** set on it, no cleanup is necessary, as the old data will have been deleted on the worker node that is no longer running the **db-1** and **www-2** pods. However, if you are using local-storage in a Kubernetes installation, then there might be some cleanup to do to remove the old data from the worker node that had the anti-affinity conflict.

Kubernetes can now reschedule the pods. All pods with persistent storage are now spread across the available worker nodes, and the pods whose PVCs were deleted will get a full copy of the data from the existing running pods. The following pod list is now observed in our example:

```
$ kubectl get po -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED	NODE
READINESS GATES								
ejs-portal-nginx-84f57ffd8c-f85wm	1/1	Running	0	30s	888.16.29.136	apimdev1066	<none>	<none>
ejs-portal-nginx-84f57ffd8c-k5klb	1/1	Running	0	103s	888.16.142.220	apimdev0129	<none>	<none>
ejs-portal-nginx-84f57ffd8c-lqhqs	1/1	Running	0	1m53s	888.16.109.212	apimdev0103	<none>	<none>
ejs-portal-sitel-db-0	2/2	Running	0	6m43s	888.16.109.211	apimdev0103	<none>	<none>
ejs-portal-sitel-db-1	2/2	Running	0	8m20s	888.16.29.134	apimdev1066	<none>	<none>
ejs-portal-sitel-db-2	2/2	Running	0	14m	888.16.142.218	apimdev0129	<none>	<none>
ejs-portal-sitel-www-0	2/2	Running	0	93s	888.16.109.213	apimdev0103	<none>	<none>
ejs-portal-sitel-www-1	2/2	Running	0	3m55s	888.16.142.219	apimdev0129	<none>	<none>
ejs-portal-sitel-www-2	2/2	Running	0	7m27s	888.16.29.135	apimdev1066	<none>	<none>
ibm-apiconnect-75b47f9f87-p25dd	1/1	Running	0	22m	888.16.109.207	apimdev0103	<none>	<none>

Migrating API test data

Back up API testing data before upgrading your IBM® API Connect to V10.0.1, and then restore the data after the upgrade is complete.

About this task

Starting with V10.0.1, the Automated API behavior testing application no longer uses MongoDB to store data. When you upgrade from V10 to V10.0.1 on Kubernetes, you must migrate data that is currently stored in MongoDB into the internal database storage used by API Connect.

You can migrate the following types of API test data:

- Snapshots
- Project environments
- Vault entities, vault variables (password fields cannot be migrated)
- Tests, test variables (test results cannot be migrated)

To migrate data, complete the following tasks:

1. [Backing up API test data](#)

Before upgrading your IBM API Connect to V10.0.1, back up your API test data from the Automated API behavior testing application.

2. [Restoring API test data](#)

After you complete the upgrade of IBM API Connect to V10.0.1, restore your API test data.

3. [Uninstalling Percona Server for MongoDB](#)

After you restore API test data in IBM API Connect V10.0.1, uninstall Percona Server for MongoDB.

Backing up API test data

Before upgrading your IBM® API Connect to V10.0.1, back up your API test data from the Automated API behavior testing application.

Open the Automated API behavior testing application and then complete the steps in the following sections to back up each type of API test data.

Snapshots

Back up each snapshot by completing the following steps:

1. Select HTTP Client.
2. Select a snapshot and load it into the client.
3. Copy the following snapshot properties to a text file:
 - Name
 - Description
 - Method
 - URL
 - Headers
 - Body
4. Save the file.

Repeat the steps for each additional snapshot that you want to migrate.

Project environments

Back up each project environment by completing the following steps:

1. Click Vault > *project name*.
2. Click Environments > Open Presets Editor.
3. Select an environment to back up.
4. Copy the following properties to a text file:
 - Name of the environment
 - Each Key:Value pair
5. Save the file.

Repeat the steps for each additional project environment that you want to migrate.

Vault entities

Back up each vault entity by completing the following steps:

1. Click Vault > *project name* > Snippets.
2. Select a snippet and copy the following properties to a text file:
 - Name
 - Description
 - Code
3. Repeat step 4 for each snippet in the vault.
4. Save the file.

Repeat this procedure for each additional vault entity that you want to migrate.

Vault variables

Back up each vault variable by completing the following steps:

1. Click Vault > *project name* > Variables.
2. Select a variable and copy the following properties to a text file:
 - Name
 - Value
3. Repeat step 2 for each variable in the vault.
4. Save the file.

Repeat the steps for each additional vault variable that you want to migrate.

Tests and test variables

Back up each test with all of its test variables in a text file by completing the following steps:

Test

1. Click Vault > *project name* > Tests.
2. Edit a test.

3. Select Source.
4. Copy the entire test document to a text file.
5. Save the file.

Test variable

To make restoring easier, use the same text file to contain a test document and the related test variables.

1. Edit the test that you just backed up.
 2. Select Data Sets.
 3. Go to a set of variables (such as **Global**) and note the scope in your file.
 4. For each variable in the data set, copy the following properties to the file containing the corresponding test document:
 - Name
 - Value
- For each data set, you should have a scope noted, plus a list of name/value pairs.
5. Repeat steps 3 and 4 for each set of variables.
 6. Save the file.

Repeat the steps for each additional test or test variable that you want to migrate.

Next topic: [Restoring API test data](#)

Restoring API test data

After you complete the upgrade of IBM® API Connect to V10.0.1, restore your API test data.

Open the Automated API behavior testing application and then complete the steps in the following sections to restore each type of API test data.

Snapshots

Restore each saved snapshot by completing the following steps:

1. Open the file containing a snapshot's data.
2. In the test application, select HTTP Client.
3. Copy the following snapshot properties into the client fields:
 - Name
 - Description
 - Method
 - URL
 - Headers
 - Body
4. Submit the request.
5. Save the snapshot.

Repeat the steps for each additional snapshot that you want to restore.

Project environments

Restore each project environment by completing the following steps:

1. Click Vault > *project name* > Tests.
2. Click New Preset.
3. Add an environment.
4. Copy the following data from the text file to the environment properties:
 - Name of the environment
 - Each Key:Value pair
5. Save the environment.

Repeat the steps for each additional project environment that you want to restore.

Vault entities

Restore each vault entity by completing the following steps to recreate the snippet:

1. Click Vault > *project name* > Tests.
2. Create a test within the project.
3. Edit the test, and select From Scratch.
4. Go to the Source view of the test and paste the snippet's code from your text file.
5. Go to the Code view and select all Components.
6. Select Vault and click the Add icon.
7. Save the snippet with the same name and description as the original.
8. Delete the test.

Repeat this procedure for each additional vault entity that you want to restore.

Vault variables

Restore each vault variable by completing the following steps:

1. Click Vault > *project name* > New Variable.
2. Paste the name and value from the file into the variable, and save it.

Repeat the steps for each additional vault variable that you want to restore.

Tests and test variables

Restore each test and its test variables by completing the following steps:

Test

1. Click Vault > *project name* > Tests.
2. Create a test with the same name as the original.
3. Select Source and paste the copied code into the source.
4. Save the new test.

Test variable

1. Select a test and click Data sets.
2. Refer to your notes and set the scope of the test variable.
3. Create the variable by copying the name and value from your file.

Repeat the steps for each additional test or test variable that you want to restore.

Previous topic: [Backing up API test data](#)

Next topic: [Uninstalling Percona Server for MongoDB](#)

Uninstalling Percona Server for MongoDB

After you restore API test data in IBM® API Connect V10.0.1, uninstall Percona Server for MongoDB.

About this task

After your API test data is migrated, you don't need Percona Server for MongoDB (PSMDB). Uninstall PSMDB by completing running the commands described in the following steps.

Procedure

1. Delete PSMDB, as well as the backup and restore CRs:

```
kubectl -n namespace delete psmdb --all
```

where *namespace* is the namespace where PSMDB was installed.

2. Delete the PSMDB Operator deployments:

```
kubectl -n namespace delete deployment percona-server-mongodb-operator
```

where *namespace* is the namespace where PSMDB was installed.

3. Delete the PSMDB service account:

```
kubectl -n namespace delete sa percona-server-mongodb-operator
```

where *namespace* is the namespace where PSMDB was installed.

4. Delete the PSMDB secrets:

```
kubectl -n namespace delete secret ibm-apiconnect-mongo-cluster-secrets
kubectl -n namespace delete secret ibm-apiconnect-mongo-cluster-ssl
kubectl -n namespace delete secret ibm-apiconnect-mongo-cluster-ssl-internal
kubectl -n namespace delete secret ibm-apiconnect-mongodb-encryption-key
```

where *namespace* is the namespace where PSMDB was installed.

5. Delete the PSMDB role binding:

```
kubectl -n namespace delete rolebindings service-account-percona-server-mongodb-operator
```

where *namespace* is the namespace where PSMDB was installed.

6. Delete the PSMDB roleDelete PSMDB role

```
kubectl -n namespace delete role
percona-server-mongodb-operator
```

where *namespace* is the namespace where PSMDB was installed.

7. Delete the PSMDB PVCs:

```
kubectl -n namespace delete pvc pvc_name
```

where:

- `namespace` is the namespace where PSMDB was installed.
- `pvc_name` is the name of a Persistent Volume Claim that is used by PSMDB.
- The PVCs use the following naming pattern: `mongod-data-ibm-apiconnect-mongo-cluster-rs0-...`

8. Delete the PSMDB CRDs:

```
kubectl delete crds perconaservermongodbackubectlups.psmdb.percona.com
kubectl delete crds perconaservermongodbrestores.psmdb.percona.com
kubectl delete crds perconaservermongodbs.psmdb.percona.com
```

Previous topic: [Restoring API test data](#)

Upgrading on OpenShift and Cloud Pak for Integration

You can upgrade API Connect in an OpenShift environment and as part of IBM Cloud Pak for Integration.

- [Upgrading API Connect in Cloud Pak for Integration](#)
In Cloud Pak for Integration, the API management capability is provided by API Connect.
- [Upgrade considerations on OpenShift](#)
Review the supported upgrade paths and other upgrade considerations on OpenShift or Cloud Pak for Integration.
- [Preparing to upgrade on OpenShift](#)
Prepare your OpenShift or Cloud Pak for Integration deployment for upgrading to the new version of API Connect 10.0.5x.
- [Upgrading on OpenShift in an online environment](#)
Perform an online (connected to the internet) upgrade of IBM® API Connect on Red Hat OpenShift Container Platform using either the top-level `APIConnectCluster` CR or individual subsystem CRs.
- [Air-gapped upgrade](#)
In a disconnected, network-restricted, upgrade API Connect on a Red Hat OpenShift Container Platform (OCP) cluster by mirroring product images to a bastion host or a portable device and then completing the upgrade process.
- [Upgrading from 2018 on OpenShift](#)
Upgrade API Connect 2018 directly to 10.0.5.x.
- [Optional post-upgrade steps for upgrade to 10.0.5.3 from earlier 10.0.5 release](#)
Version 10.0.5.3 introduces some new features that are related to inter-subsystem communication. This topic explains how to configure these features when you upgrade from a previous 10.0.5 release to v10.0.5.3 or later.
- [Enabling Analytics after upgrading](#)
After the upgrade successfully completes, re-enable Analytics data collection.
- [Troubleshooting installation and upgrade on OpenShift](#)
Review the following troubleshooting tips if you encounter a problem while installing or upgrading API Connect on OpenShift, including as a component of IBM Cloud Pak for Integration (CP4I).

Upgrading API Connect in Cloud Pak for Integration

In Cloud Pak for Integration, the API management capability is provided by API Connect.

About this task

Cloud Pak for Integration 2022 uses API Connect 10.0.5.x, which requires OpenShift 4.10 and Cloud Pak common services 3.19 (or higher). Before you can upgrade to API Connect 10.0.5.x in Cloud Pak for Integration, you must upgrade to a version of API Connect that supports the newer versions of OpenShift and common services.

Tip: After upgrade, clear your browser cache, and open a new browser window. This action avoids stale cache issues in your browser, that can result in unexpected behavior in the Cloud Manager and API Manager UIs.

Procedure

1. Required: Upgrade the API management capability to the following version of API Connect:
To ensure a successful upgrade to Cloud Pak for Integration 2022.2 with API Connect 10.0.5.x, upgrade your current version of API Connect:
 - 2020.4: If you will upgrade to Cloud Pak for Integration 2022 from 2020.4, your 2020.4 deployment must be at the latest version (API Connect 10.0.1.7-eus).
For upgrade instructions, see [Upgrading API management](#) in the 2020.4 version of the Cloud Pak for Integration documentation.
 - 2021: If you will upgrade to Cloud Pak for Integration 2022 from 2021.x, your 2021 deployment must be at the latest version of 2021.4 (API Connect 10.0.4-ifix3).
For upgrade instructions, see [Upgrading API management](#) in the 2021.4 version of the Cloud Pak for Integration documentation.

Attention: If you previously upgraded to API Connect 10.0.4-ifix3 in 2021.4, you might not have upgraded to OpenShift 4.10 because it was not required. Be sure to upgrade your deployment to OpenShift 4.10 before attempting to upgrade to API Connect 10.0.5.x in Cloud Pak for Integration 2022. For upgrade instructions, see the [Red Hat OpenShift documentation](#).
 2. **Upgrading from 10.0.1.x only:** If you installed API Connect 10.0.1.x in Cloud Pak for Integration 2020.4 using individual subsystem CRs, you must modify the CRs. This step only applies if you installed API Connect 10.0.1.x using individual subsystem CRs instead of using the instructions in the "API Management deployment" topic in the Cloud Pak for Integration documentation.
- Important: Complete this step before upgrading the `ibm-apiconnect` operator.
- a. If you don't use Platform Navigator, delete it to help avoid potential upgrade issues.
 - b. Add the following annotation to your Management, Portal, and Analytics CRs (but not the Gateway CR):

```
metadata:
  annotations:
```

```
apiconnect-operator/cp4i: "false"
```

- Upgrade the API management capability to API Connect 10.0.5.x in Cloud Pak for Integration 2022.
For instructions, see "Upgrading API management" in the appropriate version of the [Cloud Pak for Integration documentation](#).

Upgrade considerations on OpenShift

Review the supported upgrade paths and other upgrade considerations on OpenShift or Cloud Pak for Integration.

- [Supported upgrade paths](#)
- [Supported DataPower Gateway versions](#)
- [Exporting analytics data before upgrading from a version prior to 10.0.5.0](#)
- [Upgrading multiple instances of API Connect](#)
- [x-ibm-gateway-cors-allow-credentials-when-no-cors-policy deprecated](#)
- [API transactions might fail during the upgrade of gateway pods](#)
- [What to do next](#)

Tip: After upgrade, clear your browser cache, and open a new browser window. This action avoids stale cache issues in your browser, that can result in unexpected behavior in the Cloud Manager and API Manager UIs.

Supported upgrade paths

Attention: API Connect is not supported on a FIPS-enabled environment.

Table 1 lists the supported releases of IBM® API Connect v10 that can be upgraded directly to the new version of API Connect.

Table 1. Supported upgrade paths on OpenShift

Upgrade from:	How to upgrade to 10.0.5.4
<ul style="list-style-type: none">10.0.5.310.0.5.210.0.5.110.0.5.010.0.4.0- ifix310.0.1.12- eus10.0.1.11- eus10.0.1.9- eus10.0.1.8- eus10.0.1.7- eus	Complete the procedure in Upgrading on OpenShift and Cloud Pak for Integration in this documentation.
Older versions of 10.0.x: <ul style="list-style-type: none">10.0.4.0- ifix210.0.4.0- ifix110.0.4.010.0.3.0- ifix110.0.3.010.0.2.0	Complete 2 upgrades: <ol style="list-style-type: none">1. First, upgrade to 10.0.4-ifix3 as explained in Upgrading on OpenShift and Cloud Pak for Integration in the API Connect 10.0.4 documentation.2. Then, upgrade to 10.0.5.4 as explained in Upgrading on OpenShift and Cloud Pak for Integration in this documentation.

Upgrade from:	How to upgrade to 10.0.5.4
Older versions of 10.0.1.x: <ul style="list-style-type: none"> 10.0.1.6-eus 10.0.1.5-ifix4-eus 10.0.1.5-ifix3-eus 10.0.1.5-eus 10.0.1.4-ifix1-eus 10.0.1.4-eus 10.0.1.2-ifix2-eus 10.0.1.2-ifix1-eus 10.0.1.2-eus 	Complete 2 upgrades: <ol style="list-style-type: none"> First, upgrade to 10.0.1.8-eus as explained in Upgrading to 10.0.1.8-eus online in the API Connect 10.0.1.x-eus documentation. You do not need to upgrade to a higher version because you can move directly from 10.0.1.8-eus to 10.0.5.4. Then, upgrade to 10.0.5.4 as explained in Upgrading on OpenShift and Cloud Pak for Integration in this documentation.
<ul style="list-style-type: none"> 10.0.1.1 10.0.1.0 10.0.0.0-ifix2 	Complete 3 upgrades: <ol style="list-style-type: none"> First, upgrade to 10.0.1.2-ifix2-eus as explained in Upgrading on OpenShift and Cloud Pak for Integration in the API Connect 10.0.1.x-eus documentation. Note: Be sure to use the 10.0.1.2-ifix2-eus version of the operator channel, operator, and if applicable, case, as noted in the Operator, operand, and CASE version topic. Then, upgrade to 10.0.1.8-eus as explained in Upgrading to 10.0.1.8-eus online in the API Connect 10.0.1.x-eus documentation. You do not need to upgrade to a higher version because you can move directly from 10.0.1.8-eus to 10.0.5.4. Finally, upgrade to 10.0.5.4 as explained in Upgrading on OpenShift and Cloud Pak for Integration in this documentation.
2018 FP24 and higher	Upgrade to 10.0.5.4 as explained in Upgrading on OpenShift and Cloud Pak for Integration .

Attention: Different releases of API Connect support different versions of OpenShift, so you might need to upgrade OpenShift after upgrading API Connect. For more information, see [IBM API Connect Version 10 software product compatibility requirements](#).

For information on the operator, operand, and CASE version used with each API Connect release, see [Operator, operand, and CASE versions](#).

Supported DataPower Gateway versions

You can use any combination of API Connect 10.0.5.x with DataPower Gateway 10.5.0.x or DataPower API Gateway 10.5.0.x.

However, before you install, best practice is to review the latest compatibility support for your version of API Connect. To view compatibility support, follow the instructions in [IBM API Connect Version 10 software product compatibility requirements](#) to access API Connect information on the Software Product Compatibility Reports website. Once you access information for your version of API Connect, select Supported Software > Integration Middleware, and view the list of compatible DataPower Gateway versions.

Exporting analytics data before upgrading to 10.0.5.0 or later

If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher): Upgrading to 10.0.5.x from an earlier version will result in the deletion of existing analytics data. If you want to retain your analytics data then you must export it before starting the upgrade. For instructions on exporting data, see the following API Connect documentation:

- 10.0.4-ifix3: [Exporting API event data](#)
- 10.0.1.7-eus or later: [Exporting API event data](#)

Upgrading multiple instances of API Connect

Where the API Connect operator manages multiple API Connect instances, those instances should all be upgraded after upgrading the operator. Do not leave API Connect operands running on a version older than the operator for any longer than necessary to complete the upgrades, which should be done sequentially.

~~x-ibm-gateway-cors-allow-credentials-when-no-cors-policy~~ deprecated

The ~~x-ibm-gateway-cors-allow-credentials-when-no-cors-policy~~ API property is deprecated because it is replaced with the new CORS policy feature. For more information on configuring the CORS policy in v10, see the following topics:

- [Enabling CORS support for an API](#)
- OpenAPI 3 editor: [Setting API properties](#)
- OpenAPI 2 editor: [Setting API properties](#)

For information on how the new policy replaces the deprecated API property, review the following summary.

Using CORS in v2018

In v2018, when cross-origin resource sharing (CORS) is enabled for an API, every response to a CORS request contains the header:

```
Access-Control-Allow-Credentials: true
```

Inclusion of Access-Control-Allow-Credentials (ACAC) does not provide optimal security, so starting with v2018.4.1.17, an API property can be set to prevent this header from being returned:

```
Name: x-ibm-gateway-cors-allow-credentials-when-no-cors-policy
Collection: *
Value (string): off
```

The **Collection** setting refers to a specific Catalog to which a property value applies. The "*" value means the value applies to all catalogs. You can either specify * or a Catalog (collection) name.

The property can also be set by using the DataPower CLI:

```
api-definition myAPI
  property x-ibm-gateway-cors-allow-credentials-when-no-cors-policy * off
exit
```

It is only possible to turn off this header, or leave it on, for an entire API.

Important: The value of this property must be exactly the string `off` (case-sensitive) in order to remove the header.

Using CORS in V10

In API Connect 10.0.3.0 or later, the `x-ibm-gateway-cors-allow-credentials-when-no-cors-policy` property works, but only for API definitions which do NOT contain a `cors-policy`. You cannot define both this property and a CORS Policy on the same API definition.

This property is deprecated in v10: a warning will be written to the log for every CORS request received by an API whose definition contains this property.

For v10, the recommended way to ensure the ACAC header does not appear in CORS responses is to add a CORS policy to the API definition. In v10, any API definition which has a `cors-policy` defined will NOT return the ACAC header by default. Note that it is possible in the CORS policy rules to explicitly return the ACAC header for some or all origins.

If you move your configuration from Version 2018 to Version 10.0.1.4-eus or later, and have this custom property defined in your API definitions, you will not see any change in behavior, other than new warnings in the log. This property will remove the ACAC header for any API definition where it appears.

If, when you move your configuration from v2018 to v10, you want to have the ACAC header present in the CORS responses, you can leave the API definitions as-is (without the new property defined) and the ACAC header will be returned, as long as a `cors-policy` is not added to the API definition. Note also that even with a CORS policy defined for an API, the ACAC header can be returned if desired by explicitly allowing it in the rules of the CORS policy.

Note that the `x-ibm-gateway-cors-allow-credentials-when-no-cors-policy` property in an API definition:

- Will cause a commit-time error in v10 if a `cors-policy` is also present in the same API definition
- Will NOT work in v10 or v2018 to turn ON the ACAC header; it can ONLY be used to turn OFF the header. Explanation:
 - In v2018, ACAC is on by default
 - In v10, without `cors-policy`, ACAC is on by default
 - In v10, with `cors-policy`, ACAC can be turned on for specific origins (by using rules in `cors-policy`)

API transactions might fail during the upgrade of gateway pods.

Consider scheduling the upgrade during off-hours to avoid transaction failures.

When you upgrade a cluster of gateway pods in Kubernetes, a small number of API transactions may fail. During the upgrade, Kubernetes removes the pod from the load balancer configuration, deletes the pod, and then starts a new pod. The steps are repeated for each pod. Socket hang ups occur on transactions that are in process at the time the pod is deleted.

The number of transactions that fail depends on the rate of incoming transactions and the length of time needed to complete each transaction. Typically the number of failures is a very small percentage. This behavior is expected during an upgrade. If the failure level is not acceptable, schedule the upgrade during an off-hours maintenance window.

Note also that DataPower Gateway supports long-lived connections such as GraphQL subscriptions or other websockets connections. These long-lived connections might not be preserved when upgrading. Workloads with long-lived connections are more vulnerable to failed API transactions during upgrading.

You can limit the number of failed API transactions during the upgrade by using the DataPower Operator's `lifecycle` property to configure the `preStop` container lifecycle hook on the gateway pods. This approach mitigates the risk of API failures during the rolling update of the gateway StatefulSet by sleeping the pod for a span of time, allowing in-flight transactions to complete prior to the SIGTERM being delivered to the container. While this feature does not guarantee that no in-flight APIs would fail, it does provide some mitigation for in-flight transactions that can complete successfully within the configured time window. For more information, see [Delaying SIGTERM with preStop](#) in the DataPower documentation.

Upgrading to v10.0.5.3 (or later) from an earlier 10.05.x release.

Additional features related to inter-subsystem communication were added in v10.0.5.3. To enable and configure these features see: [Optional post-upgrade steps for upgrade to 10.0.5.3 from earlier 10.0.5 release](#).

What to do next

Proceed to [Preparing to upgrade on OpenShift](#) to prepare your deployment for upgrading.

- [Additional considerations for upgrading to analytics from a version prior to 10.0.5.0](#)
API Connect v10.0.5.0 introduced a new architecture for the Analytics subsystem, so some additional steps are required before and after upgrading from API Connect 10.0.4-ifix3 or 10.0.1.x.

Additional considerations for upgrading to analytics from a version prior to 10.0.5.0

API Connect v10.0.5.0 introduced a new architecture for the Analytics subsystem, so some additional steps are required before and after upgrading from API Connect 10.0.4-ifix3 or 10.0.1.x.

The key changes to the Analytics subsystem are highlighted in [What's new in Version 10.0.5](#).

The most important consideration is that existing analytics data is deleted during the upgrade from 10.0.4-ifix3, and from 10.0.1.7-eus (or higher), and cannot be restored. After upgrading to v10.0.5.0 or later, you will have an empty analytics database. If you want to keep your existing analytics data on an external system, there are three options:

- **[Exporting analytics data from the UI](#)**
Before upgrading to v10.0.5, export any analytics data that you wish to keep using the UI.
- **[Exporting analytics data by using the REST API](#)**
Before upgrading to v10.0.5, export any analytics data that you want to keep by using the REST API.
- **[Offloading analytics data](#)**
An alternative to exporting your analytics data before upgrading to v10.0.5 is to enable analytics offload. However, when enabled this feature only offloads new analytics data. But if your upgrade is not planned for a few days or weeks, you can enable this feature now and avoid the need to export any new analytics data.
- **[Remove template overrides from your analytics CR](#)**
If you are upgrading to v10.0.5.4 or previous, remove any analytics CR template overrides. v10.0.1 and v10.0.4 analytics CR template overrides are not compatible with v10.0.5 and must be removed.

Exporting analytics data from the UI

Before upgrading to v10.0.5, export any analytics data that you wish to keep using the UI.


About this task

Note: This page refers to analytics features that existed in pre-v10.0.5 releases, for the purposes of exporting analytics data before upgrade. Exporting analytics data from v10.0.5 is described here [Exporting analytics from v10.0.5](#)

Use the Analytics Discover tab to search for event data and then export the *hits* (the records in the search results) to a file. Each hit represents one API invocation in the Gateway server. When you view the hits in the UI, the display is limited to 500 records. To view the entire set of hits, export the data to a CSV or JSON Lines file.

Procedure

1. Navigate to the Analytics page and click Discover.
2. Run a search.

Select a predefined search to run, or click  to define your own search, and then run it. Optionally apply filters to refine the results. The export option is only available when a search returns at least one hit.

3. Click Export (this option displays before the search results).
4. In the "Export Hits" panel, select the Filetype for the exported data.

There are two options for the file's type:

- a. JSON Lines (.txt): Exports data to a .txt file that contains a complete JSON record for each hit.
 - b. Comma-separated Values (.csv): Exports to a .csv file that can be viewed either in a text editor, a spreadsheet, or some other CSV viewer.
- If you intend to view the .csv file in a viewer that does not support special characters, select the following options as needed to ensure compatibility with your viewer:

- i. Escape double quotes: Insert an additional quotation mark (") as an escape character before each " that is already included in the file.
- ii. Escape special characters: Insert a quotation mark (") as an escape character before each of the following characters: = _ + % @

The content of the API event records in the CSV file depends on the logging policy that is set for the operation. If the logging policy includes the HTTP headers and the payload, then these details are included in the API event record. Where the payload is large, it might take longer to complete the export of the analytics data. For more information, see [activity-log.policy](#). For information about the fields that are included in the CSV file, see [API event records](#).

5. In the panel, click Export to start the export operation.
6. When the "The export has completed successfully." message displays, click Download to save the file, or click Another one? to run a new export operation. (Remember to save the current export results before running a new export).

Attention:

- If your browser is configured to block pop-ups, the download might be blocked. The precise behavior varies across browsers, but if you see a notification about "blocked pop-ups", reconfigure the browser to always allow pop-ups for the API Manager host address. Then check the "Export Hits" panel for the Download option, which completes the operation. If you do not see a Download option, repeat the export operation.
 - If you navigate away from the Analytics interface before the exported file is downloaded, the download operation is suspended. If you return to Analytics before any other export operation begins and before the file expires, you might see a Download option that allows you to finish downloading the file. Otherwise, you must repeat the entire export operation.
 - Each downloaded file is saved to the download location that is configured for your browser, using a randomly generated file name. If you intend to export more than once, and you are exporting to the same location each time, you might want to rename each file with a meaningful name.
- **[Exporting aggregated analytics data from a visualization](#)**
From a visualization, you can view the raw aggregated data from which the graphical representation is constructed, and then export that data to a CSV file.

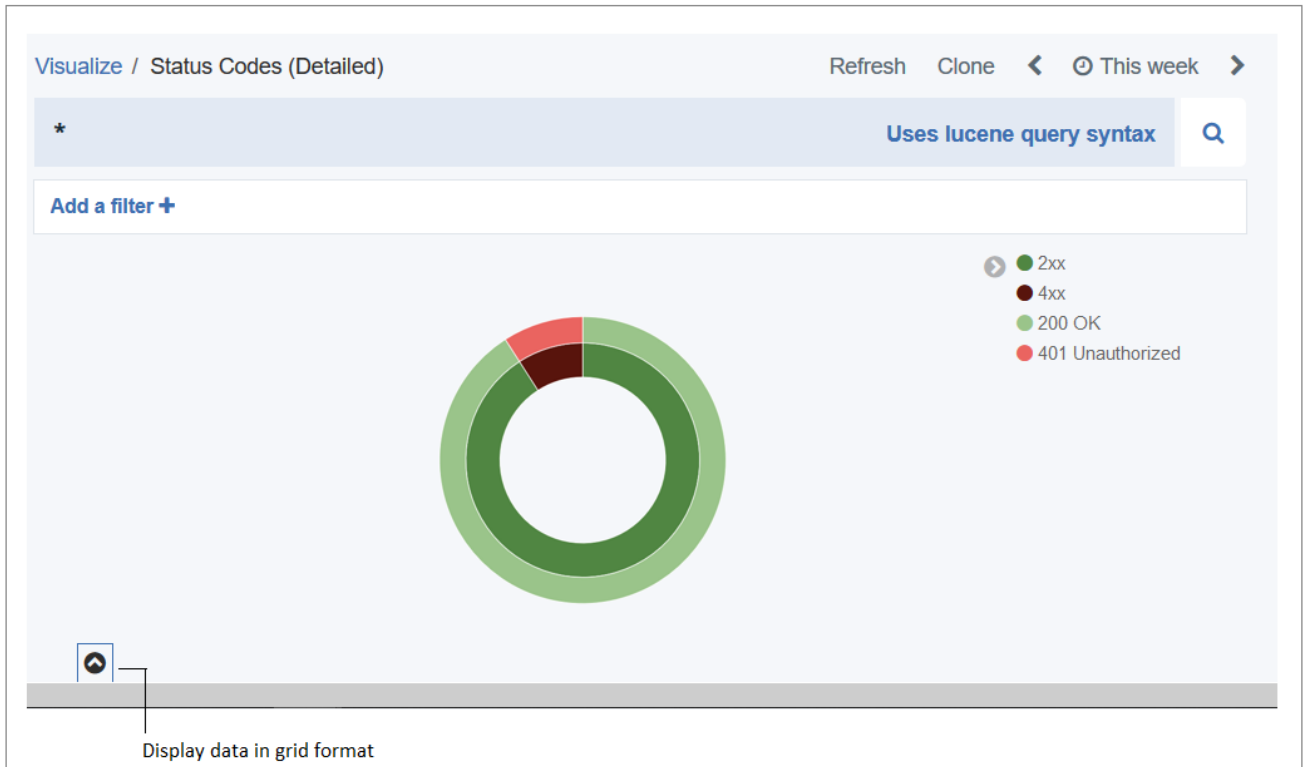
Exporting aggregated analytics data from a visualization


From a visualization, you can view the raw aggregated data from which the graphical representation is constructed, and then export that data to a CSV file.

Procedure

Note: This page refers to analytics features that existed in pre-v10.0.5 releases, for the purposes of exporting analytics data before upgrade. Exporting analytics data from v10.0.5 is described here [Exporting analytics from v10.0.5](#). To export analytics data from a visualization, complete the following steps.

1. Navigate to the Analytics page and click Visualize.
2. Select a visualization by clicking it.
For example, you might select the visualization called "Status Codes (Detailed)" as shown in the following image:



3. Display the data in grid format by clicking the  that displays after the visualization.
In the resulting grid, a row is displayed for each aggregated data set that matches the search criteria configured in the visualization's aggregation builder. The raw data shown in the columns and rows depends on the type of visualization accessed and the aggregations configured for that visualization.

Visualize / Status Codes (Detailed) Refresh Clone < This week >

* Uses lucene query syntax Q

Add a filter +

Table

filters	Count	status_code.keyword: Descending	Count
1xx	0		
2xx	20	200 OK	20
3xx	0		
4xx	2	401 Unauthorized	2
5xx	0		

Export: Raw Formatted

Page Size 10

Display in graph format Export options

4. Export a tabular format of the raw data that is displayed in the grid:
5. When prompted, choose to save the file, which is named *visualization_name.csv* by default.
The file is saved to the download location that is configured for your browser. If you intend to export more than once from the same or another visualization, and you are exporting to the same location each time, you might want to rename each file with a meaningful name.

Exporting analytics data by using the REST API

Before upgrading to v10.0.5, export any analytics data that you want to keep by using the REST API.

Before you begin

API Connect REST API calls are authenticated by using a bearer token in the authorization header. Use the toolkit credentials to request a bearer token. Toolkit credentials can be found in the API Manager UI:

1. Log in to the API Manager UI.
2. From the home page, click the Download toolkit tile.
3. Download the Toolkit credentials.
4. Open the downloaded `credentials.json` file, example:

```
...
"toolkit": {
  "endpoint": "https://mgmt.api.example.com/api",
  "client_id": "7409693f-f726-48b4-8909-7c0d26f13e81",
  "client_secret": "5feeb0be-17f8-41a4-96d6-d40f33d69ef6"
},
...
```

Take note of the `toolkit.client_id`, `toolkit.client_secret`, and `toolkit.endpoint` from this file. You will use these to get your bearer token.

About this task

The Analytics REST API is documented in the [API Connect v10.0.5LATEST](#) API Explorer. You can export the data catalog-by-catalog using `/analytics/catalogs/{org}/{catalog}/analytics-services/{analytics-service}/events`

Procedure

1. Use your provider organization credentials and `client_id/secret` to request a bearer token:

```
curl -v -k -X POST -d '{"username": "<username>", "password": "<password>", "realm": "<realm>", "client_id": "<client_id>", "client_secret": "<client_secret>", "grant_type": "password"}' -H 'Content-Type: application/json' -H
```



```
'Accept: application/json' https://<management server platform api endpoint>/token
```

Where:

- `<username>` is the provider org owner.
- `<password>` is the password for the specified `<username>`.
- `<realm>` is the user registry realm for the specified `<username>`. The default provider organization user registry realm is `provider/default-ldap-2`.
- `<client_id>` is the client ID from the "toolkit" section of the `credentials.json` file, or as provided by your API Connect cloud administrator.
- `<client_secret>` is the client secret from the "toolkit" section of the `credentials.json` file, or as provided by your API Connect cloud administrator.
- `<management server platform api endpoint>` is the platform API endpoint from the "toolkit" section of the `credentials.json` file, or as provided by your API Connect cloud administrator.

The bearer token is returned in the `access_token` property:

```
{
  "access_token": "<bearer token>",
  "token_type": "Bearer",
  "expires_in": 28800
}
```

Note the `expires_in` value, which is the number of seconds before the token expires. After expiry you need to request a new bearer token.

2. Use the returned bearer token to call the analytics REST API:

```
curl -v -k -H 'Accept: application/json' -H 'Authorization: Bearer <bearer token>' -X GET --url 'https://<management server platform api endpoint>/analytics/catalogs/<provider org name>/<catalog name>/analytics-services/<analytics service name>/events?after=now-7d'
```

Where:

- `<management server platform api endpoint>` is the `toolkit.endpoint`, but with the `/api` at the end replaced with `/analytics` to access the analytics APIs.

In this example, the REST API call returns the last 7 days (note the `?after=now-7d` at end of the URL) of event data for the specified catalog:

```
{ "total": 2, "found": true, "events": [ { "id": "AYGp1P_4umraqLKQkId3", "event": { "immediate_client_ip": "9.145.95.209", "developer_org_name": "", "resource_id": ":GET:/hola-mundo", "developer_org_id": "", "log_policy": "activity" ...
```

Note: For large data sets it might be necessary to make the REST call repeatedly, specifying a different time range for each call. For example, to capture events logged during a specific 30-minute interval, use: `?after=2022-06-28T09:00:00&before=2022-06-28T09:30:00`

3. Additional filters can be used as described in the REST API documentation for the call to `GET /analytics/catalogs/{org}/{catalog}/analytics-services/{analytics-service}/events`

Offloading analytics data

An alternative to exporting your analytics data before upgrading to v10.0.5 is to enable analytics offload. However, when enabled this feature only offloads new analytics data. But if your upgrade is not planned for a few days or weeks, you can enable this feature now and avoid the need to export any new analytics data.

The steps for configuring analytics offload are covered in the v10.0.4 documentation: <https://www.ibm.com/docs/en/api-connect/10.0.x?topic=topology-offloading-data-third-party-system>

Remove template overrides from your analytics CR

If you are upgrading to v10.0.5.4 or previous, remove any analytics CR template overrides. v10.0.1 and v10.0.4 analytics CR template overrides are not compatible with v10.0.5 and must be removed.

Note: If you are using a top-level CR, you must edit the top-level CR instead of the subsystem CR. If the section you want to edit is missing from the top-level CR, copy it from the subsystem CR and paste it into the top-level CR.

If you are upgrading to v10.0.5.4 or a previous v10.0.5 fix pack, then review edit your analytics CR and if it has a `template` section, remove this section. For example:

```
oc edit a7s -n <namespace>
```

Remove the `template` section:

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsCluster
...
spec:
...
  template: # REMOVE THIS SECTION
  - name: storage-data
    containers:
    - name: storage-data
      env:
      - name: ES_JAVA_OPTS
        value: "-Xms2g -Xmx2g"
      resources:
        requests:
          memory: 3Gi
        limits:
          memory: 4Gi
```

In this example, the template section specified custom memory allocations for internal analytics processes. These settings are applicable only to pre-v10.0.5 analytics deployments. v10.0.5 has additional deployment profiles that provide increased memory allocations: [Analytics component deployment profile limits](#).

Preparing to upgrade on OpenShift

Prepare your OpenShift or Cloud Pak for Integration deployment for upgrading to the new version of API Connect 10.0.5x.

Before you begin

Review the [Upgrade considerations on OpenShift](#) to ensure that you are following a supported upgrade path and that you understand important changes that might affect your deployment.

Attention: Upgrading to v10.0.5.x from v10.0.4-ifix3 or later, or v10.0.1.7 or later, will result in the deletion of existing analytics data. If you want to retain your analytics data then you must export it before upgrade. For instructions on exporting see [Additional considerations for upgrading analytics from a version prior to 10.0.5.0](#).

About this task

Complete the following steps to prepare your deployment for upgrading to the newest version of API Connect.

Procedure

1. If you are upgrading from API Connect 10.0.4-ifix3 and are still using OpenShift 4.6, upgrade your current deployment to OpenShift 4.10.
Attention: If you previously upgraded to API Connect 10.0.4-ifix3, you might not have upgraded OpenShift because it was not required. Be sure to upgrade your deployment to OpenShift 4.10 before attempting to upgrade to API Connect 10.0.5.x. For upgrade instructions, see the [Red Hat OpenShift documentation](#).

2. Verify that the `pgcluster` is healthy:

- a. Get the name of the `pgcluster`:

```
oc get pgcluster -n <APIC_namespace>
```

The response displays the name of the postgres cluster running in the specified namespace.

- b. Check the status of the `pgcluster`:

```
oc get pgcluster <pgcluster_name> -n <APIC_namespace> -o yaml | grep status -A 2 | tail -n3
```

The response for a healthy `pgcluster` looks like the following example, where the state is `Initialized`:

```
status:
  message: Cluster has been initialized
  state: pgcluster Initialized
```

If the cluster is healthy, proceed to the next step.

Important: If the `pgcluster` is not healthy (returns any other state):

The cluster is not healthy and an upgrade will fail. Complete the following steps:

- a. If there are any ongoing backup or restore jobs, wait until they complete and then check the status again. Do not proceed with the upgrade until the status is `Initialized`.
 - b. If all of the background jobs complete but the `pgcluster` remains unhealthy, contact IBM Support for assistance.
3. Run the pre-upgrade health check:

- a. Verify that the `apicops` utility is installed by running the following command to check the current version of the utility:

```
apicops --version
```

If the response indicates that `apicops` is not available, install it now. See [The API Connect operations tool: apicops](#) in the API Connect documentation.

- b. Run the following command to set the KUBECONFIG environment.

```
export KUBECONFIG=/path/to/kubeconfig
```

- c. Run the following command to execute the pre-upgrade script:

```
apicops version:pre-upgrade -n <namespace>
```

If the system is healthy, the results will not include any errors.

Note: This command asks for the Cloud Manager admin password in order to make a call to the platform API. If you do not have this password, or the platform API URL is not accessible from your `apicops` location, then this part of the pre-upgrade check can be skipped by adding the argument `--no-topology`. The topology check output that is produced by this API call is typically only required for debugging purposes, when the `apicops version:pre-upgrade` returns output that suggests the system is unhealthy.

4. [Back up the current deployment](#) in case the upgrade fails and you need to perform a roll back.
5. **If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher)** and you want to retain your analytics data, then you must export it before upgrade. You must also delete any `template` section from your analytics CR. For instructions on exporting data and removing the template section, see [Additional considerations for upgrading analytics from a version prior to v10.0.5.0](#)
6. **If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher)**: Disassociate and delete your Analytics services.
 - a. In Cloud Manager, click Topology.
 - b. In the section for the Availability Zone that contains the Analytics service, locate the Gateway service that the Analytics service is associated with.
 - c. Click the actions menu, and select Unassociate analytics service.
Remember to disassociate each Analytics service from all Gateways.
 - d. In the section for the Availability Zone that contains the Analytics services, locate each Analytics service and click Delete.

- Review the [Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10](#) to ensure that any customizations to the Developer Portal are compatible with Drupal 10.
In API Connect 10.0.5.3, the Developer Portal moved from Drupal 9 to Drupal 10 (this upgrade also requires PHP 8.1). The upgrade tooling will update your Developer Portal sites; however, if you have any custom modules or themes, it is your responsibility to ensure their compatibility with Drupal 10 and PHP 8.1 before starting the upgrade.

Note: If you are upgrading to 10.0.5.3 from 10.0.1.7 or earlier, or from 10.0.5.0 or earlier, your Developer Portal sites will first be upgraded to Drupal 9.5 and then to Drupal 10. When you are planning your the upgrade to 10.0.5.3, consider the extended time frame. To shorten your upgrade window, consider removing the Developer Portal sites that you do not need.

Upgrading on OpenShift in an online environment

Perform an online (connected to the internet) upgrade of IBM® API Connect on Red Hat OpenShift Container Platform using either the top-level `APICConnectCluster` CR or individual subsystem CRs.

Before you begin

- If you are upgrading an air-gapped (disconnected from the internet) installation, see [Air-gapped upgrade](#).
- The Gateway subsystem remains available during the upgrade of the Management, Portal, and Analytics subsystems.
- If you are upgrading to a version of API Connect that supports a newer version of Red Hat OpenShift, complete the API Connect upgrade before upgrading Red Hat OpenShift.
- Upgrading from 10.0.5.2 or earlier: If you did not verify that your Portal customizations are compatible with Drupal 10, do that now.
In API Connect 10.0.5.3, the Developer Portal moved from Drupal 9 to Drupal 10 (this upgrade also requires PHP 8.1). The upgrade tooling will update your Developer Portal sites; however, if you have any custom modules or themes, it is your responsibility to ensure their compatibility with Drupal 10 and PHP 8.1 before starting the upgrade. Review the [Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10](#) to ensure that any customizations to the Developer Portal are compatible with Drupal 10 and PHP 8.1.

Procedure

- Ensure that you have completed all of the steps in [Preparing to upgrade on OpenShift](#), including reviewing the [Upgrade considerations on OpenShift](#). Do not attempt an upgrade until you have reviewed the considerations and prepared your deployment.
- Ensure your API Connect deployment is ready to upgrade:
 - Your API Connect release (operand) supports a direct upgrade to this release.
For information on the operator and operand version that is used with each API Connect release, see [Operator, operand, and CASE versions](#).
 - The DataPower operator version is correct for the currently deployed version of API Connect.
For information on upgrade paths and supported versions of DataPower Gateway, see [Upgrade considerations on OpenShift](#).
 - Your deployment is running on at least the minimum supported version of Red Hat OpenShift.
For 10.0.5.4, the minimum supported version of Red Hat OpenShift of 4.10. For upgrade instructions, see the [Red Hat OpenShift documentation](#).
- Update the operator channels for DataPower and API Connect.
 - If you previously chose automatic subscriptions, the operator version upgrades automatically when you update the operator channel.
 - If you previously chose manual subscriptions and the operator channel is already on the previous version, OpenShift OLM notifies you that an upgrade is available. You must manually approve the upgrade before proceeding.
 - Both the API Connect and DataPower channels must be changed before either operator upgrades. The upgrade of both operators begins when the channel is changed for both operators.
 - a. Upgrade the DataPower operator channel to **v1.6**. From **Operators > Installed Operators**, select the **IBM DataPower Gateway** operator, then select the **Subscription** tab. Click the channel version underneath **Update channel** to update it.
Known issue:
If you see messages in the `datapower-operator` pod log indicating that the pod is waiting for a lock to be removed:

```
{"level": "info", "ts": "2021-03-08T19:29:53.432Z", "logger": "leader", "msg": "Not the leader. Waiting."}
{"level": "info", "ts": "2021-03-08T19:29:57.971Z", "logger": "leader", "msg": "Leader pod has been deleted, waiting for garbage collection to remove the lock."}
```

The DataPower operator cannot be upgraded until this problem is resolved, see: [DataPower operator documentation](#).
 - b. Update the API Connect operator channel to **v3.4**. From **Operators > Installed Operators** select the **IBM API Connect** operator, then select the **Subscription** tab. Click the channel version underneath **Update channel** to update it.
If you are upgrading from 10.0.4-ifix3 and the API Connect operator does not begin its upgrade within a few minutes, perform the following workaround to delete the `ibm-ai-wmltraining` subscription and the associated csv:
 - Run the following command to get the name of the subscription:

```
oc get subscription --no-headers=true | grep ibm-ai-wmltraining | awk '{print $1}' -n <APIC_namespace>
```
 - Run the following command to delete the subscription:

```
oc delete subscription <subscription-name> -n <APIC_namespace>
```
 - Run the following command to get the name of the csv:

```
oc get csv --no-headers=true | grep ibm-ai-wmltraining | awk '{print $1}' -n <APIC_namespace>
```
 - Run the following command to delete the csv:

```
oc delete csv <csv-name> -n <APIC_namespace>
```

Deleting the subscription and csv triggers the API Connect operator upgrade.

Wait for the operators to update, for the pods to restart, and for the instances to display the **Ready** status.

4. Ensure that the operators and operands are healthy before proceeding.

- Operators: Verify that the OpenShift UI indicates that all operators are in the **Succeeded** state without any warnings.
- If you are using a top-level CR: To verify that your API Connect cluster is healthy, run the following command:

```
oc get apiconnectcluster -n <APIC_namespace>
```

Confirm that the `apiconnectcluster` CR reports all pods as **READY**.

```
oc get ManagementCluster -n apicupgrade
NAME      READY  STATUS   VERSION   RECONCILED VERSION  AGE
management 17/17  Running  10.0.5.1  10.0.5.1-95         70m
```

- If you are using individual subsystem CRs: To verify the health of each subsystem, run the following commands:

```
oc get ManagementCluster -n <mgmt_namespace>
oc get GatewayCluster -n <gway_namespace>
oc get PortalCluster -n <portal_namespace>
oc get AnalyticsCluster -n <mgmt_namespace>
```

Check that all pods are **READY**, for example:

```
oc get PortalCluster -n apic
NAME    READY  STATUS   VERSION   RECONCILED VERSION  AGE
portal  3/3    Running  10.0.5.4  10.0.5.4-95         57m
```

5. If you are using a top-level CR: Update the top-level `apiconnectcluster` CR:

The `spec` section of the `apiconnectcluster` looks like the following example:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APICConnectCluster
metadata:
  labels:
    app.kubernetes.io/instance: apiconnect
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: apiconnect-production
  name: prod
  namespace: <APIC_namespace>
spec:
  allowUpgrade: true
  license:
    accept: true
    use: production
    license: L-GVEN-GFUPVE
  profile: n12xc4.m12
  version: 10.0.5.4
  storageClassName: rook-ceph-block
```

a. Edit the `apiconnectcluster` CR by running the following command:

```
oc -n <APIC_namespace> edit apiconnectcluster
```

b. If upgrading from **v10.0.4-ifix3**, or upgrading from **v10.0.1.7-eus (or higher)**: In the `spec` section, add a new `allowUpgrade` attribute and set it to `true`:

```
spec:
  allowUpgrade: true
```

The `allowUpgrade` attribute enables the upgrade to 10.0.5.x. Because the upgrade to 10.0.5.x deletes your analytics data, the attribute is required to prevent an accidental upgrade.

c. In the `spec` section, update the API Connect version:

Change the `version` setting to `10.0.5.4`.

d. If you are upgrading to a version of API Connect that requires a new license, update the license value now.

For the list of licenses, see [API Connect licenses](#).

e. In the `spec.gateway` section of the CR, delete any `template` or `dataPowerOverride` sections.

You cannot perform an upgrade if the CR contains an override.

f. Save and close the CR to apply your changes.

The response looks like the following example:

```
apiconnectcluster.apiconnect.ibm.com/prod configured
```

Note: If you see an error message when you attempt to save the CR, check if it is one of the following known issues:

- Webhook error for incorrect license.
If you did not update the license ID in the CR, then when you save your changes, the following webhook error might display:

```
admission webhook "vapiconnectcluster.kb.io" denied the request:
APICConnectCluster.apiconnect.ibm.com "<instance-name>" is invalid: spec.license.license:
Invalid value: "L-RJON-BYGHM4": License L-RJON-BYGHM4 is invalid for the chosen version version.
Please refer license document https://ibm.biz/apiclicenses
```

To resolve the error, see [API Connect licenses](#) for the list of the available license IDs and select the appropriate license IDs for your deployment. Update the CR with the new license value as in the following example, and then save and apply your changes again.

- Webhook error: **Original PostgreSQL primary not found**. Take the following actions to complete the upgrade and fix the cause of the error message:

- Edit your `apiconnectcluster` CR and add the following annotation:

```

...
metadata:
  annotations:
    apiconnect-operator/db-primary-not-found-allow-upgrade=true
...

```

- Continue with the upgrade. When the upgrade is complete, the management CR reports the warning:

```
Original PostgreSQL primary not found, perform db backup and restore to restore original primary.
```

- Take a new [management database backup](#).
 - Immediately [restore](#) from the new backup taken in the previous step. The action of taking and restoring a management backup results in the establishment of a new Postgres primary, eliminating the CR warning message. Be careful to restore from the backup that is taken after the upgrade, and not from a backup taken before upgrade.
 - Webhook error: `Original postgres primary is running as replica`. Complete a Postgres failover, see [Postgres failover steps](#). After you apply the Postgres failover steps, the upgrade resumes automatically.
- g. Run the following command to verify that the upgrade is completed and the status of the top-level CR is **READY**:

```
oc get apiconnectcluster -n <APIC_namespace>
```

Important: If you need to restart the deployment, wait until all Portal sites complete the upgrade. Run the following commands to check the status of the sites:

- Get the portal service ID and endpoint:

```
apic portal-services:get -o admin -s <management_server_endpoint> \
--availability-zone <availability-zone-default> <portal-service-name> \
--output - --format json
```

- List the sites:

```
apic --mode portaladmin sites:list -s <management_server_endpoint> \
--portal_service_name <portal-service-name> \
--format json
```

Any sites currently upgrading display the **UPGRADING** status; any site that completed its upgrade displays the **INSTALLED** status and the new platform version. Verify that all sites display the **INSTALLED** status before proceeding.

For more information on the `sites` command, see [apic sites:list](#) and [Using the sites commands](#).

- After all sites are in **INSTALLED** state and have the new platform listed, run:

```
apic --mode portaladmin platforms:list -s <management_server_endpoint> \
--portal_service_id <portal_service_id_from_above_command> \
--portal_service_endpoint <portal_service_endpoint_from_above_command> \
--format json
```

Verify that the new version of the platform is the only platform listed.

For more information on the `platforms` command, see [apic platforms:list](#) and [Using the platforms commands](#).

6. If you are using individual subsystem CRs: Start with the Management subsystem. Update the Management CR as follows:

- Edit the `ManagementCluster` CR:

```
oc edit ManagementCluster -n <mgmt_namespace>
```

- If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher):** In the `spec` section, add a new `allowUpgrade` attribute and set it to `true`

```
spec:
  allowUpgrade: true
```

The `allowUpgrade` attribute enables the upgrade to 10.0.5.x. Because the upgrade to 10.0.5.x. deletes your analytics data, the attribute is required to prevent an accidental upgrade.

- In the `spec` section, update the API Connect version:
Change the `version` setting to `10.0.5.4`.
 - If you are upgrading to a version of API Connect that requires a new license, update the license value now.
For the list of licenses, see [API Connect licenses](#).
- Save and close the CR to apply your changes.
The response looks like the following example:

```
managementcluster.management.apiconnect.ibm.com/management edited
```

Note: If you see an error message when you attempt to save the CR, check if it is one of the following known issues:

- Webhook error for incorrect license.
If you did not update the license ID in the CR, then when you save your changes, the following webhook error might display:

```
admission webhook "vapiconnectcluster.kb.io" denied the request:
APIConnectCluster.apiconnect.ibm.com "<instance-name>" is invalid: spec.license.license:
Invalid value: "L-RJON-BYGHM4": License L-RJON-BYGHM4 is invalid for the chosen version version.
Please refer license document https://ibm.biz/apiclicenses
```

To resolve the error, see [API Connect licenses](#) for the list of the available license IDs and select the appropriate license IDs for your deployment. Update the CR with the new license value as in the following example, and then save and apply your changes again.

- Webhook error: `Original PostgreSQL primary not found`. Take the following actions to complete the upgrade and fix the cause of the error message:
 - Edit your `ManagementCluster` CR and add the following annotation:

```

...
metadata:

```

```

annotations:
  apiconnect-operator/db-primary-not-found-allow-upgrade=true
  ...

```

- Continue with the upgrade. When the upgrade is complete, the management CR reports the warning:

```
Original PostgreSQL primary not found, perform db backup and restore to restore original primary.
```

- Take a new [management database backup](#).
 - Immediately [restore](#) from the new backup taken in the previous step. The action of taking and restoring a management backup results in the establishment of a new Postgres primary, eliminating the CR warning message. Be careful to restore from the backup that is taken after the upgrade, and not from a backup taken before upgrade.
 - Webhook error: **Original postgres primary is running as replica**. Complete a Postgres failover, see [Postgres failover steps](#). After you apply the Postgres failover steps, the upgrade resumes automatically.
- f. Wait until the Management subsystem upgrade is complete before proceeding to the next subsystem. Check the status of the upgrade with: `oc get ManagementCluster -n <mgmt_namespace>`, and wait until all pods are running at the new version. For example:

```

oc get ManagementCluster -n <mgmt_namespace>
NAME      READY   STATUS    VERSION      RECONCILED VERSION  AGE
management 18/18   Running  10.0.5.4     10.0.5.4-1281      97m

```

- g. Repeat the process for the remaining subsystem CRs in your preferred order: **GatewayCluster**, **PortalCluster**, **AnalyticsCluster**.

Important:

- In the **GatewayCluster** CR, delete any `template` or `dataPowerOverride` sections. You cannot perform an upgrade if the CR contains an override.
 - **If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher):** The `allowUpgrade` attribute set in the Management CR must also be set in the **AnalyticsCluster** CR. It is not required for Gateway or Portal CRs.
- h. If you need to restart the deployment, wait until all Portal sites complete the upgrade. Run the following commands to check the status of the sites:

- i. Log in as an admin user:

```
apic login -s <server_name> --realm admin/default-idp-1 --username admin --password <password>
```

- ii. Get the portal service ID and endpoint:

```
apic portal-services:get -o admin -s <management_server_endpoint> \
  --availability-zone availability-zone-default <portal-service-name> \
  --output - --format json
```

- iii. List the sites:

```
apic --mode portaladmin sites:list -s <management_server_endpoint> \
  --portal_service_name <portal-service-name> \
  --format json
```

Any sites currently upgrading display the **UPGRADING** status; any site that completed its upgrade displays the **INSTALLED** status and the new platform version. Verify that all sites display the **INSTALLED** status before proceeding.

For more information on the `sites` command, see [apic sites:list](#) and [Using the sites commands](#).

- iv. After all sites are in **INSTALLED** state and have the new platform listed, run:

```
apic --mode portaladmin platforms:list -s <server_name> --portal_service_name <portal_service_name>
```

Verify that the new version of the platform is the only platform listed.

For more information on the `platforms` command, see [apic platforms:list](#) and [Using the platforms commands](#).

- If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher):** Enable analytics as explained in [Enabling Analytics after upgrading](#)
- Upgrade to Red Hat OpenShift Container Platform 4.12 if you have not already done so. Red Hat OpenShift requires you to upgrade in stages, so that you install every version between your starting point and your ending point. For example, to upgrade from 4.10 to 4.12, you must complete 2 upgrades:
 - Upgrade to 4.11
 - Upgrade to 4.12
 For upgrade instructions, see the [Red Hat OpenShift documentation](#).
- If you are upgrading to 10.0.5.3 (or later) from an earlier 10.0.5.x release: Review and configure the new inter-subsystem communication features: [Optional post-upgrade steps for upgrade to 10.0.5.3 from earlier 10.0.5 release](#).
- Restart all nats-server pods by running the following command:

```
oc -n <namespace> delete po -l app.kubernetes.io/name=natscluster
```

What to do next

Update your toolkit CLI by downloading it from IBM Fix Central or from the Cloud Manager UI, see [Installing the toolkit](#).

If you are upgrading from v10.0.5.1 or earlier to v10.0.5.2: The change in deployment profile CPU and memory limits that are introduced in 10.0.5.2 (see [New deployment profiles and CPU licensing](#)) can result in a change in the performance of your Management component. If you notice any obvious reduction in performance of the Management UI or toolkit CLI where you have multiple concurrent users, open a support case.

Air-gapped upgrade

In a disconnected, network-restricted, upgrade API Connect on a Red Hat OpenShift Container Platform (OCP) cluster by mirroring product images to a bastion host or a portable device and then completing the upgrade process.

About this task

If you previously deployed or upgraded API Connect in an air-gapped environment, you can use the same bastion host, portable computer, or portable storage as before and just update the product information for the new release.

- **Upgrading with a bastion host**
Use a bastion host to perform an air-gapped upgrade of IBM® API Connect on Red Hat OpenShift Container Platform (OCP) using either the top-level **APIConnectCluster** CR or individual subsystem CRs. First the operator is updated, and then the operands (IBM API Connect itself).
- **Upgrading with a portable computer or storage device**
Use a portable computer or portable storage device to perform an air-gapped upgrade of IBM API Connect on Red Hat OpenShift Container Platform (OCP) using either the top-level **APIConnectCluster** CR or individual subsystem CRs. First the operator is updated, and then the operands (IBM API Connect itself).

Upgrading with a bastion host

Use a bastion host to perform an air-gapped upgrade of IBM® API Connect on Red Hat OpenShift Container Platform (OCP) using either the top-level **APIConnectCluster** CR or individual subsystem CRs. First the operator is updated, and then the operands (IBM API Connect itself).

Before you begin

- If you are upgrading an installation online (connected to the internet), see [Upgrading on OpenShift in an online environment](#).
- The upgrade procedure requires you to use Red Hat Skopeo for moving container images. Skopeo is not available for Microsoft Windows, so you cannot perform this task using a Windows host.
- If you are upgrading to a version of API Connect that supports a newer version of Red Hat OpenShift, complete the API Connect upgrade before upgrading Red Hat OpenShift.
- Upgrading from 10.0.5.2 or earlier: If you did not verify that your Portal customizations are compatible with Drupal 10, do that now. In API Connect 10.0.5.3, the Developer Portal moved from Drupal 9 to Drupal 10 (this upgrade also requires PHP 8.1). The upgrade tooling will update your Developer Portal sites; however, if you have any custom modules or themes, it is your responsibility to ensure their compatibility with Drupal 10 and PHP 8.1 before starting the upgrade. Review the [Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10](#) to ensure that any customizations to the Developer Portal are compatible with Drupal 10 and PHP 8.1.

About this task

- The Gateway subsystem remains available during the upgrade of the Management, Portal, and Analytics subsystems.
- Don't use the tilde ~ within double quotation marks in any command because the tilde doesn't expand and your commands might fail.

Procedure

1. Ensure that you have completed all of the steps in [Preparing to upgrade on OpenShift](#), including reviewing the [Upgrade considerations on OpenShift](#). Do not attempt an upgrade until you have reviewed the considerations and prepared your deployment.
2. Set up the mirroring environment.
 - a. Prepare the target cluster:
 - Deploy a supported version of Red Hat OpenShift Container Platform (OCP) as a cluster.
For information, see Table 2 "API Connect and OpenShift Container Platform (OCP) compatibility matrix" in [IBM API Connect Version 10 software product compatibility requirements](#).
 - Configure storage on the cluster and make sure that it is available.
 - b. Prepare the bastion host:
You must be able to connect your bastion host to the internet and to the restricted network environment (with access to the Red Hat OpenShift Container Platform (OCP) cluster and the local registry) at the same time. Your host must be on a Linux x86_64 or Mac platform with any operating system that the Red Hat OpenShift Client supports (in Windows, execute the actions in a Linux x86_64 VM or from a Windows Subsystem for Linux terminal).

- i. Ensure that the sites and ports listed in Table 1 can be reached from the bastion host:

Table 1. Sites that must be reached from the bastion host

Site	Description
<code>icr.io:443</code>	IBM entitled registry
<code>quay.io:443</code>	Local API Connect image repository
<code>github.com</code>	CASE files and tools
<code>redhat.com</code>	Red Hat OpenShift Container Platform (OCP) upgrades

- ii. On the bastion host, install either Docker or Podman (not both).
Docker and Podman are used for managing containers; you only need to install one of these applications.
 - To install Docker (for example, on Red Hat Enterprise Linux), run the following commands:

```
yum check-update
yum install docker
```
 - To install Podman, see the [Podman installation instructions](#).
For example, on Red Hat Enterprise Linux 9, install Podman with the following command:

```
yum install podman
```
- iii. Install the Red Hat OpenShift Client tool (oc) as explained in [Getting started with the OpenShift CLI](#).
The oc tool is used for managing Red Hat OpenShift resources in the cluster.
- iv. Download the IBM Catalog Management Plug-in for IBM Cloud Paks version 1.1.0 or later from [GitHub](#).

The **ibm-pak** plug-in enables you to access hosted product images, and to run **oc ibm-pak** commands against the cluster. To confirm that **ibm-pak** is installed, run the following command and verify that the response lists the command usage:

```
oc ibm-pak --help
```

c. Set up a local image registry and credentials.

The local Docker registry stores the mirrored images in your network-restricted environment.

i. Install a registry, or get access to an existing registry.

You might already have access to one or more centralized, corporate registry servers to store the API Connect images. If not, then you must install and configure a production-grade registry before proceeding.

The registry product that you use must meet the following requirements:

- Supports multi-architecture images through Docker Manifest V2, Schema 2
For details, see [Docker Manifest V2, Schema 2](#).
- Is accessible from the Red Hat OpenShift Container Platform cluster nodes
- Allows path separators in the image name

Note: Do not use the Red Hat OpenShift image registry as your local registry because it does not support multi-architecture images or path separators in the image name.

ii. Configure the registry to meet the following requirements:

- Supports auto-repository creation
- Has sufficient storage to hold all of the software that is to be transferred
- Has the credentials of a user who can create and write to repositories (the mirroring process uses these credentials)
- Has the credentials of a user who can read all repositories (the Red Hat OpenShift Container Platform cluster uses these credentials)

To access your registries during an air-gapped installation, use an account that can write to the target local registry. To access your registries during runtime, use an account that can read from the target local registry.

3. Set environment variables and download CASE files.

Create environment variables to use while mirroring images, connect to the internet, and download the API Connect CASE files.

a. Create the following environment variables with the installer image name and the image inventory on your host:

```
export CASE_NAME=ibm-apiconnect
export CASE_VERSION=4.0.5
export ARCH=amd64
```

For information on API Connect CASE versions and their corresponding operators and operands, see [Operator, operand, and CASE versions](#).

b. Connect your host to the internet (it does not need to be connected to the network-restricted environment at this time).

c. Download the CASE file to your host:

```
oc ibm-pak get $CASE_NAME --version $CASE_VERSION
```

If you omit the **--version** parameter, the command downloads the latest version.

4. Mirror the images.

The process of mirroring images pulls the images from the internet and pushes them to your local registry. After mirroring your images, you can configure your cluster and pull the images to it before installing API Connect.

a. Generate mirror manifests.

i. Define the environment variable **\$TARGET_REGISTRY** by running the following command:

```
export TARGET_REGISTRY=<target-registry>
```

Replace **<target-registry>** with the IP address (or host name) and port of the local registry; for example: **172.16.0.10:5000**. If you want the images to use a specific namespace within the target registry, you can specify it here; for example: **172.16.0.10:5000/registry_ns**.

ii. Generate mirror manifests by running the following commands:

```
oc ibm-pak generate mirror-manifests $CASE_NAME --version $CASE_VERSION $TARGET_REGISTRY
```

If you need to filter for a specific image group, add the parameter **--filter <image_group>** to this command.

The **generate** command creates the following files at **~/ibm-pak/data/mirror/\$CASE_NAME/\$CASE_VERSION** and **~/ibm-pak/data/mirror/\$CS_CASE_NAME/\$CS_CASE_VERSION**:

- catalog-sources.yaml
- catalog-sources-linux-<arch>.yaml (if there are architecture-specific catalog sources)
- image-content-source-policy.yaml
- images-mapping.txt

The files are used when mirroring the images to the **TARGET_REGISTRY**.

b. Obtain an entitlement key for the entitled registry where the images are hosted:

- Log in to the [IBM Container Library](#).
- In the Container software library, select Get entitlement key.
- In the "Access your container software" section, click Copy key.
- Copy the key to a safe location; you will use it to log in to **cp.icr.io** in the next step.

c. Authenticate with the entitled registry where the images are hosted.

The image pull secret allows you to authenticate with the entitled registry and access product images.

i. Run the following command to export the path to the file that will store the authentication credentials that are generated on a Podman or Docker login:

```
export REGISTRY_AUTH_FILE=$HOME/.docker/config.json
```

The authentication file is typically located at **\$HOME/.docker/config.json** on Linux or **%USERPROFILE%/.docker/config.json** on Windows.

ii. Log in to the **cp.icr.io** registry with Podman or Docker; for example:


```
podman login cp.icr.io
```

Use `cp` as the username and your entitlement key as the password.

d. Authenticate with the local registry.

Log in to the local registry using an account that can write images to that registry; for example:

```
podman login $TARGET_REGISTRY
```

If the registry is insecure, add the following flag to the command: `--tls-verify=false`.

e. Update the CASE manifest to correctly reference the DataPower Operator image.

Files for the DataPower Operator are now hosted on `icr.io`; however, the CASE manifest still refers to `docker.io` as the image host. To work around this issue, visit [Airgap install failure due to 'unable to retrieve source image docker.io'](#) in the DataPower documentation and update the manifest as instructed. After the manifest is updated, continue to the next step in this procedure.

f. Mirror the product images.

- i. Connect the bastion host to both the internet and the restricted-network environment that contains the local registry.
- ii. Run the following commands to copy the images to the local registry:

```
oc image mirror \  
-f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping.txt \  
-a $REGISTRY_AUTH_FILE \  
--filter-by-os '*' \  
--skip-multiple-scopes \  
--max-per-registry=1
```

Note: If the local registry is not secured by TLS, or the certificate presented by the local registry is not trusted by your device, add the `--insecure` option to the command.

There might be a slight delay before you see a response to the command.

g. Configure the target cluster.

Now that images have been mirrored to the local registry, the target cluster must be configured to pull the images from it. Complete the following steps to configure the cluster's global pull secret with the local registry's credentials and then instruct the cluster to pull the images from the local registry.

- i. Log in to your Red Hat OpenShift Container Platform cluster:

```
oc login <openshift_url> -u <username> -p <password> -n <namespace>
```

- ii. [Update the global image pull secret](#) for the cluster as explained in the Red Hat OpenShift Container Platform documentation. Updating the image pull secret provides the cluster with the credentials needed for pulling images from your local registry.

Note: If you have an insecure registry, add the registry to the cluster's `insecureRegistries` list by running the following command:

```
oc edit image.config.openshift.io/cluster -o yaml
```

and add the `TARGET_REGISTRY` to `spec.registrySources.insecureRegistries` as shown in the following example:

```
spec:  
  registrySources:  
    insecureRegistries:  
      - insecure0.svc:5001  
      - <TARGET_REGISTRY>
```

If the `insecureRegistries` field does not exist, you can add it.

- iii. Create the `ImageContentSourcePolicy`, which instructs the cluster to pull the images from your local registry (run both commands):

```
oc apply -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/image-content-source-policy.yaml
```

- iv. Verify that the `ImageContentSourcePolicy` resource was created:

```
oc get imageContentSourcePolicy
```

- v. Verify your cluster node status:

```
oc get MachineConfigPool -w
```

Wait for all nodes to be updated before proceeding to the next step.

5. Apply the catalog sources.

Now that you have mirrored images to the target cluster, apply the catalog sources.

In the following steps, replace `<Architecture>` with either `amd64`, `s390x` or `ppc64le` as appropriate for your environment.

- a. Export the variables for the command line to use:

```
export CASE_NAME=ibm-apiconnect  
export CASE_VERSION=4.0.5  
export ARCH=amd64
```

- b. Generate the catalog sources and save them in another directory in case you need to replicate this installation in the future.

- i. Get the catalog sources:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
```

- ii. Get any architecture-specific catalog sources that you need to back up as well:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

You can also navigate to the directory in your file browser to copy these artifacts into files that you can keep for re-use or for pipelines.

- c. Apply the catalog sources to the cluster.

- i. Apply the universal catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
```

- ii. Apply any architecture-specific catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

- iii. Confirm that the catalog sources have been created in the `openshift-marketplace` namespace:

```
oc get catalogsource -n openshift-marketplace
```

6. Update the operator channel.

- a. Open the Red Hat OpenShift web console and click Operators > Installed Operators > IBM API connect > Subscriptions.

- b. Change the channel to the new version (v3.4), which triggers an upgrade of the API Connect operator.

If you are upgrading from 10.0.4-ifix3 and the API Connect operator does not begin its upgrade within a few minutes, perform the following workaround to delete the `ibm-ai-wmltraining` subscription and associated csv:

- i. Run the following command to get the name of the subscription:

```
oc get subscription -n <APIC_namespace> --no-headers=true | grep ibm-ai-wmltraining | awk '{print $1}'
```

- ii. Run the following command to delete the subscription:

```
oc delete subscription <subscription-name> -n <APIC_namespace>
```

- iii. Run the following command to get the name of the csv:

```
oc get csv --no-headers=true -n <APIC_namespace> | grep ibm-ai-wmltraining | awk '{print $1}'
```

- iv. Run the following command to delete the csv:

```
oc delete csv <csv-name> -n <APIC_namespace>
```

Deleting the subscription and csv triggers the API Connect operator upgrade.

When the API Connect operator is updated, the new operator pod starts automatically.

7. Verify that the API Connect operator was updated by completing the following steps:

- a. Get the name of the pod that hosts the operator by running the following command:

```
oc get po -n <APIC_namespace> | grep apiconnect
```

The response looks like the following example:

```
ibm-apiconnect-7bdb795465-8f7rm          1/1      Running    0          4m23s
```

- b. Get the API Connect version deployed on that pod by running the following command:

```
oc describe po <ibm-apiconnect-operator-podname> -n <APIC_namespace> | grep -i productversion
```

The response looks like the following example:

```
productVersion: 10.0.5.4
```

8. If you are using a top-level CR: Update the top-level `APIConnectCluster` CR:

The `spec` section of the `apiconnectcluster` looks like the following example:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APIConnectCluster
metadata:
  labels:
    app.kubernetes.io/instance: apiconnect
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: apiconnect-production
  name: prod
  namespace: <APIC_namespace>
spec:
  allowUpgrade: true
  license:
    accept: true
    use: production
    license: L-GVEN-GFUPVE
  profile: n12xc4.m12
  version: 10.0.5.4
  storageClassName: rook-ceph-block
```

- a. Edit the `apiconnectcluster` CR by running the following command:

```
oc -n <APIC_namespace> edit apiconnectcluster
```

- b. If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher): In the `spec` section, add a new `allowUpgrade` attribute and set it to `true`:

```
spec:
  allowUpgrade: true
```

The `allowUpgrade` attribute enables the upgrade to 10.0.5.x. Because the upgrade deletes your analytics data, the attribute is required to prevent an accidental upgrade.

- c. In the `spec` section, update the API Connect version:

Change the `version` setting to `10.0.5.4`.

- d. In the `spec.gateway` section of the CR, delete any `template` or `dataPowerOverride` sections.

You cannot perform an upgrade if the CR contains an override.

e. Save and close the CR.

The response looks like the following example:

```
apiconnectcluster.apiconnect.ibm.com/prod configured
```

Note: If you see an error message when you attempt to save the CR, check if it is one of the following known issues:

- Webhook error for incorrect license.

If you did not update the license ID in the CR, then when you save your changes, the following webhook error might display:

```
admission webhook "vapiconnectcluster.kb.io" denied the request:
APIConnectCluster.apiconnect.ibm.com "<instance-name>" is invalid: spec.license.license:
Invalid value: "L-RJON-BYGHM4": License L-RJON-BYGHM4 is invalid for the chosen version version.
Please refer license document https://ibm.biz/apiclicenses
```

To resolve the error, see [API Connect licenses](#) for the list of the available license IDs and select the appropriate license IDs for your deployment.

Update the CR with the new license value as in the following example, and then save and apply your changes again.

- Webhook error: **Original PostgreSQL primary not found**. Take the following actions to complete the upgrade and fix the cause of the error message:

- Edit your `apiconnectcluster` CR and add the following annotation:

```
...
metadata:
  annotations:
    apiconnect-operator/db-primary-not-found-allow-upgrade=true
...
```

- Continue with the upgrade. When the upgrade is complete, the management CR reports the warning:

```
Original PostgreSQL primary not found, perform db backup and restore to restore original primary.
```

- Take a new [management database backup](#).
- Immediately [restore](#) from the new backup taken in the previous step. The action of taking and restoring a management backup results in the establishment of a new Postgres primary, eliminating the CR warning message. Be careful to restore from the backup that is taken after the upgrade, and not from a backup taken before upgrade.
- Webhook error: **Original postgres primary is running as replica**. Complete a Postgres failover, see [Postgres failover steps](#). After you apply the Postgres failover steps, the upgrade resumes automatically.

9. If you are using individual subsystem CRs: Start with the Management subsystem and update by completing the following steps:

a. Edit the `ManagementCluster` CR:

```
oc edit ManagementCluster -n <mgmt_namespace>
```

b. **If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher):** In the `spec` section, add a new `allowUpgrade` attribute and set it to `true`.

```
spec:
  allowUpgrade: true
```

The `allowUpgrade` attribute enables the upgrade to 10.0.5.x. Because the upgrade to 10.0.5.x. deletes your analytics data, the attribute is required to prevent an accidental upgrade.

c. In the `spec` section, update the API Connect version:

Change the `version` setting to `10.0.5.4`.

d. If you are upgrading to a version of API Connect that requires a new license, update the license value now.

For the list of licenses, see [API Connect licenses](#).

e. Save and close the CR to apply your changes.

The response looks like the following example:

```
managementcluster.management.apiconnect.ibm.com/management edited
```

Note: If you see an error message when you attempt to save the CR, check if it is one of the following known issues:

- Webhook error for incorrect license.

If you did not update the license ID in the CR, then when you save your changes, the following webhook error might display:

```
admission webhook "vapiconnectcluster.kb.io" denied the request:
APIConnectCluster.apiconnect.ibm.com "<instance-name>" is invalid: spec.license.license:
Invalid value: "L-RJON-BYGHM4": License L-RJON-BYGHM4 is invalid for the chosen version version.
Please refer license document https://ibm.biz/apiclicenses
```

To resolve the error, see [API Connect licenses](#) for the list of the available license IDs and select the appropriate license IDs for your deployment.

Update the CR with the new license value as in the following example, and then save and apply your changes again.

- Webhook error: **Original PostgreSQL primary not found**. Take the following actions to complete the upgrade and fix the cause of the error message:

- Edit your `ManagementCluster` CR and add the following annotation:

```
...
metadata:
  annotations:
    apiconnect-operator/db-primary-not-found-allow-upgrade=true
...
```

- Continue with the upgrade. When the upgrade is complete, the management CR reports the warning:

```
Original PostgreSQL primary not found, perform db backup and restore to restore original primary.
```

- Take a new [management database backup](#).
- Immediately [restore](#) from the new backup taken in the previous step. The action of taking and restoring a management backup results in the establishment of a new Postgres primary, eliminating the CR warning message. Be careful to restore from the backup that is taken after the upgrade, and not from a backup taken before upgrade.

- Webhook error: Original postgres primary is running as replica. Complete a Postgres failover, see [Postgres failover steps](#). After you apply the Postgres failover steps, the upgrade resumes automatically.
- f. Confirm that the Management subsystem upgrade is complete before proceeding to the next subsystem.

Check the status of the upgrade with: `oc get ManagementCluster -n <mgmt_namespace>`, and wait until all pods are running at the new version. For example:

```
oc get ManagementCluster -n <mgmt_namespace>
NAME      READY   STATUS    VERSION    RECONCILED VERSION  AGE
management 18/18   Running  10.0.5.4  10.0.5.4-1281      97m
```

- g. Repeat the process for the remaining subsystem CRs: `GatewayCluster`, `PortalCluster`, and then `AnalyticsCluster`.

Important:

- In the `GatewayCluster` CR, delete any `template` or `dataPowerOverride` sections. You cannot perform an upgrade if the CR contains an override.
- **If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher):** The `allowUpgrade` attribute set in the Management CR must also be set in the `AnalyticsCluster` CR. It is not required for Gateway or Portal CRs.

10. Validate that the upgrade was successfully deployed by running the following command:

```
oc get apic -n <APIC_namespace>
```

The response looks like the following example:

```
NAME      READY   STATUS    VERSION    RECONCILED VERSION  AGE
analyticscluster.analytics.apiconnect.ibm.com/prod-a7s 5/5   Running  10.0.5.4  10.0.5.4  21h

NAME      READY   STATUS    VERSION    RECONCILED VERSION  AGE
apiconnectcluster.apiconnect.ibm.com/prod 4/4   Ready    10.0.5.4  10.0.5.4  22h

NAME      PHASE    READY   SUMMARY          VERSION  AGE
datapowerservice.datapower.ibm.com/prod-gw Running  True    StatefulSet replicas ready: 3/3 10.0.5.4 21h

NAME      PHASE    LAST EVENT    WORK PENDING    WORK IN-PROGRESS  AGE
datapowermonitor.datapower.ibm.com/prod-gw Running  false         false            21h

NAME      READY   STATUS    VERSION    RECONCILED VERSION  AGE
gatewaycluster.gateway.apiconnect.ibm.com/prod-gw 2/2   Running  10.0.5.4  10.0.5.4  21h

NAME      READY   STATUS    VERSION    RECONCILED VERSION  AGE
managementcluster.management.apiconnect.ibm.com/prod-mgmt 16/16 Running  10.0.5.4  10.0.5.4  22h

NAME      TYPE    CR TYPE    AGE    STATUS    ID          CLUSTER
managementbackup.management.apiconnect.ibm.com/prod-mgmt-0f583bd9 incr record 11h    Ready    20210505-141020F_20210506-011830I prod-mgmt
managementbackup.management.apiconnect.ibm.com/prod-mgmt-10af02ee full record 21h    Ready    20210505-141020F prod-mgmt
managementbackup.management.apiconnect.ibm.com/prod-mgmt-148f0cfa incr record 11h    Ready    20210505-141020F_20210506-012856I prod-mgmt
managementbackup.management.apiconnect.ibm.com/prod-mgmt-20bd6dae incr record 3h28m  Ready    20210505-141020F_20210506-090753I prod-mgmt
managementbackup.management.apiconnect.ibm.com/prod-mgmt-40efdb38 incr record 16h    Ready    20210505-141020F_20210505-195838I prod-mgmt
managementbackup.management.apiconnect.ibm.com/prod-mgmt-681aa239 incr record 14h    Ready    20210505-141020F_20210505-220302I prod-mgmt
managementbackup.management.apiconnect.ibm.com/prod-mgmt-7f7150dd incr record 20h    Ready    20210505-141020F_20210505-160732I prod-mgmt
managementbackup.management.apiconnect.ibm.com/prod-mgmt-806f8de6 incr record 14h    Ready    20210505-141020F_20210505-214657I prod-mgmt
managementbackup.management.apiconnect.ibm.com/prod-mgmt-868a066a incr record 3h34m  Ready    20210505-141020F_20210506-090140I prod-mgmt
managementbackup.management.apiconnect.ibm.com/prod-mgmt-cf9a85dc incr record 15h    Ready    20210505-141020F_20210505-210119I prod-mgmt
managementbackup.management.apiconnect.ibm.com/prod-mgmt-ef63b789 full record 83m    Ready    20210506-103241F prod-mgmt

NAME      STATUS    MESSAGE
managementdbupgrade.management.apiconnect.ibm.com/prod-mgmt-up-649mc Complete Upgrade is Complete (DB Schema/data are up-to-date) 142m
managementdbupgrade.management.apiconnect.ibm.com/prod-mgmt-up-9mjhk Complete Fresh install is Complete (DB Schema/data are up-to-date) 22h

NAME      READY   STATUS    VERSION    RECONCILED VERSION  AGE
portalcluster.portal.apiconnect.ibm.com/prod-ptl 3/3   Running  10.0.5.4  10.0.5.4  21h
```

Important: If you need to restart the deployment, wait until all Portal sites complete the upgrade. Run the following commands to check the status of the sites:

- a. Log in as an admin user:

```
apic login -s <server_name> --realm admin/default-idp-1 --username admin --password <password>
```

- b. Get the portal service ID and endpoint:

```
apic portal-services:get -o admin -s <management_server_endpoint> \
--availability-zone availability-zone-default <portal-service-name> \
--output - --format json
```

- c. List the sites:

```
apic --mode portaladmin sites:list -s <management_server_endpoint> \
--portal_service_name <portal-service-name> \
--format json
```

Any sites currently upgrading display the **UPGRADING** status; any site that completed its upgrade displays the **INSTALLED** status and the new platform version. Verify that all sites display the **INSTALLED** status before proceeding.

For more information on the `sites` command, see [apic sites:list](#) and [Using the sites commands](#).

- d. After all sites are in **INSTALLED** state and have the new platform listed, run the following command:

```
apic --mode portaladmin platforms:list -s <server_name> --portal_service_name <portal_service_name>
```

Verify that the new version of the platform is the only platform listed.

For more information on the `platforms` command, see [apic platforms:list](#) and [Using the platforms commands](#).

11. **If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher):** Enable analytics as explained in [Enabling Analytics after upgrading](#)
12. Upgrade to Red Hat OpenShift Container Platform 4.12 if you have not already done so.
Red Hat OpenShift requires you to upgrade in stages, so that you install every version between your starting point and your ending point. For example, to upgrade from 4.10 to 4.12, you must complete 2 upgrades:
 - a. Upgrade to 4.11
 - b. Upgrade to 4.12For upgrade instructions, see the [Red Hat OpenShift documentation](#).
13. If you are upgrading to 10.0.5.3 (or later) from an earlier 10.0.5.x release: Review and configure the new inter-subsystem communication features: [Optional post-upgrade steps for upgrade to 10.0.5.3 from earlier 10.0.5 release](#).
14. Restart all nats-server pods by running the following command:

```
oc -n <namespace> delete po -l app.kubernetes.io/name=natscluster
```

What to do next

Update your toolkit CLI by downloading it from IBM Fix Central or from the Cloud Manager UI, see [Installing the toolkit](#).

If you are upgrading from v10.0.5.1 or earlier to v10.0.5.2: The change in deployment profile CPU and memory limits that are introduced in 10.0.5.2 (see [New deployment profiles and CPU licensing](#)) can result in a change in the performance of your Management component. If you notice any obvious reduction in performance of the Management UI or toolkit CLI where you have multiple concurrent users, open a support case.

Upgrading with a portable computer or storage device

Use a portable computer or portable storage device to perform an air-gapped upgrade of IBM® API Connect on Red Hat OpenShift Container Platform (OCP) using either the top-level **APICConnectCluster** CR or individual subsystem CRs. First the operator is updated, and then the operands (IBM API Connect itself).

Before you begin

- If you are upgrading an installation online (connected to the internet), see [Upgrading on OpenShift in an online environment](#).
- The upgrade procedure requires you to use Red Hat Skopeo for moving container images. Skopeo is not available for Microsoft Windows, so you cannot perform this task using a Windows host.
- If you are upgrading to a version of API Connect that supports a newer version of Red Hat OpenShift, complete the API Connect upgrade before upgrading Red Hat OpenShift.
- Upgrading from 10.0.5.2 or earlier: If you did not verify that your Portal customizations are compatible with Drupal 10, do that now.
In API Connect 10.0.5.3, the Developer Portal moved from Drupal 9 to Drupal 10 (this upgrade also requires PHP 8.1). The upgrade tooling will update your Developer Portal sites; however, if you have any custom modules or themes, it is your responsibility to ensure their compatibility with Drupal 10 and PHP 8.1 before starting the upgrade. Review the [Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10](#) to ensure that any customizations to the Developer Portal are compatible with Drupal 10 and PHP 8.1.

About this task

- The Gateway subsystem remains available during the upgrade of the Management, Portal, and Analytics subsystems.
- Don't use the tilde ~ within double quotation marks in any command because the tilde doesn't expand and your commands might fail.

Procedure

1. Ensure that you have completed all of the steps in [Preparing to upgrade on OpenShift](#), including reviewing the [Upgrade considerations on OpenShift](#). Do not attempt an upgrade until you have reviewed the considerations and prepared your deployment.
2. Set up the mirroring environment.
 - a. Prepare the target cluster:
 - Deploy a supported version of Red Hat OpenShift Container Platform (OCP) as a cluster.
For information, see Table 2 "API Connect and OpenShift Container Platform (OCP) compatibility matrix" in [IBM API Connect Version 10 software product compatibility requirements](#).
 - Configure storage on the cluster and make sure that it is available.
 - b. Prepare the portable device:
You must be able to connect your portable device to the internet and to the restricted network environment (with access to the Red Hat OpenShift Container Platform (OCP) cluster and the local registry). The portable device must be on a Linux x86_64 or Mac platform with any operating system that the Red Hat OpenShift Client supports (in Windows, execute the actions in a Linux x86_64 VM or from a Windows Subsystem for Linux terminal).
 - i. Ensure that the portable device has sufficient storage to hold all of the software that is to be transferred to the local registry.
 - ii. On the portable device, install either Docker or Podman (not both).

Docker and Podman are used for managing containers; you only need to install one of these applications.

- To install Docker (for example, on Red Hat Enterprise Linux), run the following commands:

```
yum check-update
yum install docker
```

- To install Podman, see the [Podman installation instructions](#).
For example, on Red Hat Enterprise Linux 9, install Podman with the following command:

```
yum install podman
```

- iii. Install the Red Hat OpenShift Client tool (`oc`) as explained in [Getting started with the OpenShift CLI](#).
The `oc` tool is used for managing Red Hat OpenShift resources in the cluster.

- iv. Download the IBM Catalog Management Plug-in for IBM Cloud Paks version 1.1.0 or later from [GitHub](#).
The `ibm-pak` plug-in enables you to access hosted product images, and to run `oc ibm-pak` commands against the cluster. To confirm that `ibm-pak` is installed, run the following command and verify that the response lists the command usage:

```
oc ibm-pak --help
```

- c. Set up a local image registry and credentials.

The local Docker registry stores the mirrored images in your network-restricted environment.

- i. Install a registry, or get access to an existing registry.

You might already have access to one or more centralized, corporate registry servers to store the API Connect images. If not, then you must install and configure a production-grade registry before proceeding.

The registry product that you use must meet the following requirements:

- Supports multi-architecture images through Docker Manifest V2, Schema 2
For details, see [Docker Manifest V2, Schema 2](#).
- Is accessible from the Red Hat OpenShift Container Platform cluster nodes
- Allows path separators in the image name

Note: Do not use the Red Hat OpenShift image registry as your local registry because it does not support multi-architecture images or path separators in the image name.

- ii. Configure the registry to meet the following requirements:

- Supports auto-repository creation
- Has sufficient storage to hold all of the software that is to be transferred
- Has the credentials of a user who can create and write to repositories (the mirroring process uses these credentials)
- Has the credentials of a user who can read all repositories (the Red Hat OpenShift Container Platform cluster uses these credentials)

To access your registries during an air-gapped installation, use an account that can write to the target local registry. To access your registries during runtime, use an account that can read from the target local registry.

3. Set environment variables and download CASE files.

Create environment variables to use while mirroring images, connect to the internet, and download the API Connect CASE files.

- a. Create the following environment variables with the installer image name and the image inventory on your portable device:

```
export CASE_NAME=ibm-apiconnect
export CASE_VERSION=4.0.5
export ARCH=amd64
```

For information on API Connect CASE versions and their corresponding operators and operands, see [Operator, operand, and CASE versions](#).

- b. Connect your portable device to the internet (it does not need to be connected to the network-restricted environment at this time).
- c. Download the CASE file to your portable device:

```
oc ibm-pak get $CASE_NAME --version $CASE_VERSION
```

If you omit the `--version` parameter, the command downloads the latest version.

4. Mirror the images.

The process of mirroring images pulls the images from the internet and pushes them to your local registry. After mirroring your images, you can configure your cluster and pull the images to it before installing API Connect.

- a. Generate mirror manifests.

- i. Define the environment variable `$TARGET_REGISTRY` by running the following command:

```
export TARGET_REGISTRY=<target-registry>
```

Replace `<target-registry>` with the IP address (or host name) and port of the local registry; for example: `172.16.0.10:5000`. If you want the images to use a specific namespace within the target registry, you can specify it here; for example: `172.16.0.10:5000/registry_ns`.

- ii. Generate mirror manifests by running the following commands:

```
oc ibm-pak generate mirror-manifests $CASE_NAME file:///integration --version $CASE_VERSION --final-registry $TARGET_REGISTRY
```

If you need to filter for a specific image group, add the parameter `--filter <image_group>` to this command.

The `generate` command creates the following files at `~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION`:

- `catalog-sources.yaml`
- `catalog-sources-linux-<arch>.yaml` (if there are architecture-specific catalog sources)
- `image-content-source-policy.yaml`
- `images-mapping-to-filesystem.txt`
- `images-mapping-from-filesystem.txt`

The files are used when mirroring the images to the `TARGET_REGISTRY`.

- b. Obtain an entitlement key for the entitled registry where the images are hosted:
 - i. Log in to the [IBM Container Library](#).
 - ii. In the Container software library, select Get entitlement key.
 - iii. In the "Access your container software" section, click Copy key.
 - iv. Copy the key to a safe location; you will use it to log in to `cp.icr.io` in the next step.

c. Authenticate with the entitled registry where the images are hosted.

The image pull secret allows you to authenticate with the entitled registry and access product images.

- i. Run the following command to export the path to the file that will store the authentication credentials that are generated on a Podman or Docker login:

```
export REGISTRY_AUTH_FILE=$HOME/.docker/config.json
```

The authentication file is typically located at `$HOME/.docker/config.json` on Linux or `%USERPROFILE%\.docker/config.json` on Windows.

- ii. Log in to the `cp.icr.io` registry with Podman or Docker; for example:

```
podman login cp.icr.io
```

Use `cp` as the username and your entitlement key as the password.

- d. Update the API Connect CASE manifest to correctly reference the DataPower Operator image.

Files for the DataPower Operator are now hosted on `icr.io`; however, the CASE manifest still refers to `docker.io` as the image host. To work around this issue, visit [Airgap install failure due to 'unable to retrieve source image docker.io'](#) in the DataPower documentation and update the manifest as instructed. The manifest for API Connect (which uses the DataPower Operator) is stored in `~/ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION`.

After the manifest is updated, continue to the next step in this procedure.

- e. Mirror the images from the internet to the portable device.

- i. Define the environment variable `$IMAGE_PATH` by running the following command:

```
export IMAGE_PATH=<image-path>
```

where `<image-path>` indicates where the files will be stored on the portable device's file system.

- ii. Mirror the images to the portable device:

```
oc image mirror \
-f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping-to-filesystem.txt \
--filter-by-os '*' \
-a $REGISTRY_AUTH_FILE \
--skip-multiple-scopes \
--max-per-registry=1 \
--dir "$IMAGE_PATH"
```

There might be a slight delay before you see a response to the command.

- f. Move the portable device to the restricted-network environment.

The procedure depends on the type of device that you are using:

If you are using a portable computer, disconnect the device from the internet and connect it to the restricted-network environment. The same environment variables can be used.

If you are using portable storage, complete the following steps:

- i. Transfer the following files to a device in the restricted-network environment:
 - The `~/ibm-pak` directory.
 - The contents of the `<image-path>` that you specified in the previous step.
- ii. Create the same environment variables as on the original device; for example:

```
export CASE_NAME=ibm-apiconnect
export CASE_VERSION=4.0.5
export ARCH=amd64

export REGISTRY_AUTH_FILE=$HOME/.docker/config.json

export IMAGE_PATH=<image-path>
```

- g. Authenticate with the local registry.

Log in to the local registry using an account that can write images to that registry; for example:

```
podman login $TARGET_REGISTRY
```

If the registry is insecure, add the following flag to the command: `--tls-verify=false`.

- h. Mirror the product images to the target registry.

- i. If you are using a portable computer, connect it to the restricted-network environment that contains the local registry. If you are using portable storage, you already transferred files to a device within the restricted-network environment.
- ii. Run the following commands to copy the images to the local registry:

```
oc image mirror \
-f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping-from-filesystem.txt \
-a $REGISTRY_AUTH_FILE \
--filter-by-os '*' \
--skip-multiple-scopes \
--max-per-registry=1 \
--from-dir "$IMAGE_PATH"
```

Note: If the local registry is not secured by TLS, or the certificate presented by the local registry is not trusted by your device, add the `--insecure` option to the command.

There might be a slight delay before you see a response to the command.

- i. Configure the target cluster.

Now that images have been mirrored to the local registry, the target cluster must be configured to pull the images from it. Complete the following steps to configure the cluster's global pull secret with the local registry's credentials and then instruct the cluster to pull the images from the local registry.

- i. Log in to your Red Hat OpenShift Container Platform cluster:

```
oc login <openshift_url> -u <username> -p <password> -n <namespace>
```

- ii. [Update the global image pull secret](#) for the cluster as explained in the Red Hat OpenShift Container Platform documentation. Updating the image pull secret provides the cluster with the credentials needed for pulling images from your local registry.

Note: If you have an insecure registry, add the registry to the cluster's `insecureRegistries` list by running the following command:

```
oc edit image.config.openshift.io/cluster -o yaml
```

and add the `TARGET_REGISTRY` to `spec.registrySources.insecureRegistries` as shown in the following example:

```
spec:
  registrySources:
    insecureRegistries:
      - insecure0.svc:5001
      - <TARGET_REGISTRY>
```

If the `insecureRegistries` field does not exist, you can add it.

- iii. Create the `ImageContentSourcePolicy`, which instructs the cluster to pull the images from your local registry (run both commands):

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/image-content-source-policy.yaml
```

- iv. Verify that each `ImageContentSourcePolicy` resource was created:

```
oc get imageContentSourcePolicy
```

- v. Verify your cluster node status:

```
oc get MachineConfigPool -w
```

Wait for all nodes to be updated before proceeding to the next step.

5. Apply the catalog sources.

Now that you have mirrored images to the target cluster, apply the catalog sources.

In the following steps, replace `<Architecture>` with either `amd64`, `s390x` or `ppc64le` as appropriate for your environment.

- a. Export the variables for the command line to use:

```
export CASE_NAME=ibm-apiconnect
export CASE_VERSION=4.0.5
export ARCH=amd64
```

- b. Generate the catalog source and save it in another directory in case you need to replicate this installation in the future.

- i. Get the catalog source:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
```

- ii. Get any architecture-specific catalog sources that you need to back up as well:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

You can also navigate to the directory in your file browser to copy these artifacts into files that you can keep for re-use or for pipelines.

- c. Apply the catalog sources to the cluster.

- i. Apply the universal catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
```

- ii. Apply any architecture-specific catalog source:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

- iii. Confirm that the catalog source was created in the `openshift-marketplace` namespace:

```
oc get catalogsource -n openshift-marketplace
```

6. Update the operator channel.

- a. Open the Red Hat OpenShift web console and click `Operators`, `Installed Operators`, `IBM API connect`, `Subscriptions`.

- b. Change the channel to the new version (`v3.4`), which triggers an upgrade of the API Connect operator.

If you are upgrading from `10.0.4-ifix3` and the API Connect operator does not begin its upgrade within a few minutes, perform the following workaround to delete the `ibm-ai-wmltraining` subscription and associated csv:

- i. Run the following command to get the name of the subscription:

```
oc get subscription -n <APIC_namespace> --no-headers=true | grep ibm-ai-wmltraining | awk '{print $1}'
```

- ii. Run the following command to delete the subscription:

```
oc delete subscription <subscription-name> -n <APIC_namespace>
```

- iii. Run the following command to get the name of the csv:

```
oc get csv --no-headers=true -n <APIC_namespace> | grep ibm-ai-wmltraining | awk '{print $1}'
```

- iv. Run the following command to delete the csv:

```
oc delete csv <csv-name> -n <APIC_namespace>
```


Deleting the subscription and csv triggers the API Connect operator upgrade.

When the API Connect operator is updated, the new operator pod starts automatically.

7. Verify that the API Connect operator was updated by completing the following steps:

a. Get the name of the pod that hosts the operator by running the following command:

```
oc get po -n <APIC_namespace> | grep apiconnect
```

The response looks like the following example:

```
ibm-apiconnect-7bdb795465-8f7rm          1/1      Running    0          4m23s
```

b. Get the API Connect version deployed on that pod by running the following command:

```
oc describe po <ibm-apiconnect-operator-podname> -n <APIC_namespace> | grep -i productversion
```

The response looks like the following example:

```
productVersion: 10.0.5.4
```

8. If you are using a top-level CR: Update the top-level `APIConnectCluster` CR:

The `spec` section of the `apiconnectcluster` looks like the following example:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APIConnectCluster
metadata:
  labels:
    app.kubernetes.io/instance: apiconnect
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: apiconnect-production
  name: prod
  namespace: <APIC_namespace>
spec:
  allowUpgrade: true
  license:
    accept: true
    use: production
    license: L-GVEN-GFUPVE
  profile: n12xc4.m12
  version: 10.0.5.4
  storageClassName: rook-ceph-block
```

a. Edit the `apiconnectcluster` CR by running the following command:

```
oc -n <APIC_namespace> edit apiconnectcluster
```

b. If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher): In the `spec` section, add a new `allowUpgrade` attribute and set it to `true`:

```
spec:
  allowUpgrade: true
```

The `allowUpgrade` attribute enables the upgrade to 10.0.5.x. Because the upgrade deletes your analytics data, the attribute is required to prevent an accidental upgrade.

c. In the `spec` section, update the API Connect version:

Change the `version` setting to `10.0.5.4`.

d. In the `spec.gateway` section of the CR, delete any `template` or `dataPowerOverride` sections.

You cannot perform an upgrade if the CR contains an override.

e. Save and close the CR.

The response looks like the following example:

```
apiconnectcluster.apiconnect.ibm.com/prod configured
```

Note: If you see an error message when you attempt to save the CR, check if it is one of the following known issues:

- Webhook error for incorrect license.

If you did not update the license ID in the CR, then when you save your changes, the following webhook error might display:

```
admission webhook "vapiconnectcluster.kb.io" denied the request:
APIConnectCluster.apiconnect.ibm.com "<instance-name>" is invalid: spec.license.license:
Invalid value: "L-RJON-BYGHM4": License L-RJON-BYGHM4 is invalid for the chosen version.
Please refer license document https://ibm.biz/apiclicenses
```

To resolve the error, see [API Connect licenses](#) for the list of the available license IDs and select the appropriate license IDs for your deployment.

Update the CR with the new license value as in the following example, and then save and apply your changes again.

- Webhook error: `Original PostgreSQL primary not found`. Take the following actions to complete the upgrade and fix the cause of the error message:

- Edit your `apiconnectcluster` CR and add the following annotation:

```
...
metadata:
  annotations:
    apiconnect-operator/db-primary-not-found-allow-upgrade=true
...
```

- Continue with the upgrade. When the upgrade is complete, the management CR reports the warning:

```
Original PostgreSQL primary not found, perform db backup and restore to restore original primary.
```

- Take a new [management database backup](#).

- Immediately [restore](#) from the new backup taken in the previous step. The action of taking and restoring a management backup results in the establishment of a new Postgres primary, eliminating the CR warning message. Be careful to restore from the backup that is taken after the upgrade, and not from a backup taken before upgrade.
 - Webhook error: **Original postgres primary is running as replica**. Complete a Postgres failover, see [Postgres failover steps](#). After you apply the Postgres failover steps, the upgrade resumes automatically.
9. If you are using individual subsystem CRs: Start with the Management subsystem and update by completing the following steps:
- a. Edit the **ManagementCluster** CR:

```
oc edit ManagementCluster -n <mgmt_namespace>
```

- b. **If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher):** In the `spec` section, add a new `allowUpgrade` attribute and set it to `true`.

```
spec:
  allowUpgrade: true
```

The `allowUpgrade` attribute enables the upgrade to 10.0.5.x. Because the upgrade to 10.0.5.x deletes your analytics data, the attribute is required to prevent an accidental upgrade.

- c. In the `spec` section, update the API Connect version:
Change the `version` setting to `10.0.5.4`.
- d. If you are upgrading to a version of API Connect that requires a new license, update the license value now.
For the list of licenses, see [API Connect licenses](#).

- e. Save and close the CR to apply your changes.
The response looks like the following example:

```
managementcluster.management.apiconnect.ibm.com/management edited
```

Note: If you see an error message when you attempt to save the CR, check if it is one of the following known issues:

- Webhook error for incorrect license.
If you did not update the license ID in the CR, then when you save your changes, the following webhook error might display:

```
admission webhook "vapiconnectcluster.kb.io" denied the request:
APIConnectCluster.apiconnect.ibm.com "<instance-name>" is invalid: spec.license.license:
Invalid value: "L-RJON-BYGHM4": License L-RJON-BYGHM4 is invalid for the chosen version version.
Please refer license document https://ibm.biz/apiclicenses
```

To resolve the error, see [API Connect licenses](#) for the list of the available license IDs and select the appropriate license IDs for your deployment. Update the CR with the new license value as in the following example, and then save and apply your changes again.

- Webhook error: **Original PostgreSQL primary not found**. Take the following actions to complete the upgrade and fix the cause of the error message:
 - Edit your **ManagementCluster** CR and add the following annotation:

```
...
metadata:
  annotations:
    apiconnect-operator/db-primary-not-found-allow-upgrade=true
...
```

- Continue with the upgrade. When the upgrade is complete, the management CR reports the warning:
Original PostgreSQL primary not found, perform db backup and restore to restore original primary.
- Take a new [management database backup](#).
- Immediately [restore](#) from the new backup taken in the previous step. The action of taking and restoring a management backup results in the establishment of a new Postgres primary, eliminating the CR warning message. Be careful to restore from the backup that is taken after the upgrade, and not from a backup taken before upgrade.
- Webhook error: **Original postgres primary is running as replica**. Complete a Postgres failover, see [Postgres failover steps](#). After you apply the Postgres failover steps, the upgrade resumes automatically.

- f. Confirm that the Management subsystem upgrade is complete before proceeding to the next subsystem.

Check the status of the upgrade with: `oc get ManagementCluster -n <mgmt_namespace>`, and wait until all pods are running at the new version. For example:

```
oc get ManagementCluster -n <mgmt_namespace>
NAME          READY   STATUS    VERSION   RECONCILED VERSION   AGE
management    18/18   Running   10.0.5.4   10.0.5.4-1281        97m
```

- g. Repeat the process for the remaining subsystem CRs: **GatewayCluster**, **PortalCluster**, and then **AnalyticsCluster**.

Important:

- In the **GatewayCluster** CR, delete any `template` or `dataPowerOverride` sections. You cannot perform an upgrade if the CR contains an override.
- **If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher):** The `allowUpgrade` attribute set in the Management CR must also be set in the **AnalyticsCluster** CR. It is not required for Gateway or Portal CRs.

10. Validate that the upgrade was successfully deployed by running the following command:

```
oc get apic -n <APIC_namespace>
```

The response looks like the following example:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
analyticscluster.analytics.apiconnect.ibm.com/prod-a7s	5/5	Running	10.0.5.4	10.0.5.4		21h

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
apiconnectcluster.apiconnect.ibm.com/prod	4/4	Ready	10.0.5.4	10.0.5.4		22h

NAME	PHASE	READY	SUMMARY	VERSION	AGE
datapowerservice.datapower.ibm.com/prod-gw	Running	True	StatefulSet replicas ready: 3/3	10.0.5.4	21h

NAME	PHASE	LAST EVENT	WORK PENDING	WORK IN-PROGRESS	AGE
------	-------	------------	--------------	------------------	-----

```

datapowermonitor.datapower.ibm.com/prod-gw Running false false 21h
NAME READY STATUS VERSION RECONCILED VERSION AGE
gatewaycluster.gateway.apiconnect.ibm.com/prod-gw 2/2 Running 10.0.5.4 10.0.5.4 21h
NAME READY STATUS VERSION RECONCILED VERSION
AGE
managementcluster.management.apiconnect.ibm.com/prod-mgmt 16/16 Running 10.0.5.4 10.0.5.4 22h
NAME STATUS ID CLUSTER
TYPE CR TYPE AGE
managementbackup.management.apiconnect.ibm.com/prod-mgmt-0f583bd9 Ready 20210505-141020F_20210506-011830I prod-mgmt
incr record 11h
managementbackup.management.apiconnect.ibm.com/prod-mgmt-10af02ee Ready 20210505-141020F prod-mgmt
full record 21h
managementbackup.management.apiconnect.ibm.com/prod-mgmt-148f0cfa Ready 20210505-141020F_20210506-012856I prod-mgmt
incr record 11h
managementbackup.management.apiconnect.ibm.com/prod-mgmt-20bd6dae Ready 20210505-141020F_20210506-090753I prod-mgmt
incr record 3h28m
managementbackup.management.apiconnect.ibm.com/prod-mgmt-40efdb38 Ready 20210505-141020F_20210505-195838I prod-mgmt
incr record 16h
managementbackup.management.apiconnect.ibm.com/prod-mgmt-681aa239 Ready 20210505-141020F_20210505-220302I prod-mgmt
incr record 14h
managementbackup.management.apiconnect.ibm.com/prod-mgmt-7f7150dd Ready 20210505-141020F_20210505-160732I prod-mgmt
incr record 20h
managementbackup.management.apiconnect.ibm.com/prod-mgmt-806f8de6 Ready 20210505-141020F_20210505-214657I prod-mgmt
incr record 14h
managementbackup.management.apiconnect.ibm.com/prod-mgmt-868a066a Ready 20210505-141020F_20210506-090140I prod-mgmt
incr record 3h34m
managementbackup.management.apiconnect.ibm.com/prod-mgmt-cf9a85dc Ready 20210505-141020F_20210505-210119I prod-mgmt
incr record 15h
managementbackup.management.apiconnect.ibm.com/prod-mgmt-ef63b789 Ready 20210506-103241F prod-mgmt
full record 83m
NAME STATUS MESSAGE
AGE
managementdbupgrade.management.apiconnect.ibm.com/prod-mgmt-up-649mc Complete Upgrade is Complete (DB Schema/data are
up-to-date) 142m
managementdbupgrade.management.apiconnect.ibm.com/prod-mgmt-up-9mjhk Complete Fresh install is Complete (DB
Schema/data are up-to-date) 22h
NAME READY STATUS VERSION RECONCILED VERSION AGE
portalcluster.portal.apiconnect.ibm.com/prod-ptl 3/3 Running 10.0.5.4 10.0.5.4 21h

```

Important: If you need to restart the deployment, wait until all Portal sites complete the upgrade. Run the following commands to check the status of the sites:

- a. Log in as an admin user:

```
apic login -s <server_name> --realm admin/default-idp-1 --username admin --password <password>
```

- b. Get the portal service ID and endpoint:

```
apic portal-services:get -o admin -s <management_server_endpoint> \
--availability-zone availability-zone-default <portal-service-name> \
--output - --format json
```

- c. List the sites:

```
apic --mode portaladmin sites:list -s <management_server_endpoint> \
--portal_service_name <portal-service-name> \
--format json
```

Any sites currently upgrading display the **UPGRADING** status; any site that completed its upgrade displays the **INSTALLED** status and the new platform version. Verify that all sites display the **INSTALLED** status before proceeding.

For more information on the `sites` command, see [apic sites:list](#) and [Using the sites commands](#).

- d. After all sites are in **INSTALLED** state and have the new platform listed, run the following command:

```
apic --mode portaladmin platforms:list -s <server_name> --portal_service_name <portal_service_name>
```

Verify that the new version of the platform is the only platform listed.

For more information on the `platforms` command, see [apic platforms:list](#) and [Using the platforms commands](#).

11. If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher): Enable analytics as explained in [Enabling Analytics after upgrading](#)
12. Upgrade to Red Hat OpenShift Container Platform 4.12 if you have not already done so. Red Hat OpenShift requires you to upgrade in stages, so that you install every version between your starting point and your ending point. For example, to upgrade from 4.10 to 4.12, you must complete 2 upgrades:
 - a. Upgrade to 4.11
 - b. Upgrade to 4.12

For upgrade instructions, see the [Red Hat OpenShift documentation](#).
13. If you are upgrading to 10.0.5.3 (or later) from an earlier 10.0.5.x release: Review and configure the new inter-subsystem communication features: [Optional post-upgrade steps for upgrade to 10.0.5.3 from earlier 10.0.5 release](#).
14. Restart all nats-server pods by running the following command:

```
oc -n <namespace> delete po -l app.kubernetes.io/name=natscluster
```

What to do next

Update your toolkit CLI by downloading it from IBM Fix Central or from the Cloud Manager UI, see [Installing the toolkit](#).

If you are upgrading from v10.0.5.1 or earlier to v10.0.5.2: The change in deployment profile CPU and memory limits that are introduced in 10.0.5.2 (see [New deployment profiles and CPU licensing](#)) can result in a change in the performance of your Management component. If you notice any obvious reduction in performance of the Management UI or toolkit CLI where you have multiple concurrent users, open a support case.

Upgrading from 2018 on OpenShift

Upgrade API Connect 2018 directly to 10.0.5.x.

About this task

Complete the API Connect upgrade from 2018 to 10.0.5.x by performing the following tasks:

1. [OpenShift: Preparing the 2018 deployment for upgrade](#)
Prepare the API Connect V2018 deployment for the upgrade to the latest version 10.0.5.x on OpenShift.
2. [OpenShift: Upgrading from 2018 to v10](#)
Perform an online upgrade of IBM® API Connect 2018 to the latest release of v10 on OpenShift using a top-level upgrade CR.
3. [OpenShift: Post-upgrade steps](#)
After you complete the API Connect upgrade, remove files left over from Version 2018 and optionally enable the Automated API behavior testing application.
4. [OpenShift: Recovering the V2018 deployment](#)
Use the disaster recovery process to revert a failed upgrade to the IBM API Connect V2018 deployment on OpenShift.

OpenShift: Preparing the 2018 deployment for upgrade

Prepare the API Connect V2018 deployment for the upgrade to the latest version 10.0.5.x on OpenShift.

Before you begin

Review the IBM Support page [API Connect: v2018 to v10 upgrade paths](#) to determine whether your Version 2018 deployment can be directly upgraded to the latest release of API Connect.

Procedure

1. Review the 10.0.5.x requirements and features; in particular:
 - [IBM API Connect Version 10 software product compatibility requirements](#)
 - [Deployment requirements](#)
 - [Planning your deployment](#)
 - [What's New in the latest release](#)
 - [Version 10 known limitations](#)
2. Configure external logging prior to upgrade. This is not mandatory but is recommended. For example, <https://logdna.com/>.
3. Be sure you know the name of each subsystem on your 2018 deployment.
The 2018 subsystem names are carried over to 10.0.5.x installations. To view the names, run the following command:

```
apicup subsys list
```

These subsystem names are the values used in the `spec.subsystemName` field for each custom resource (CR) when upgrading.

4. On 2018, check the health of each subsystem before upgrade:

```
apicup subsys health-check <subsystem_name>
```

5. On the 2018 system, back up the original `apicup` project directory to a location from which it can always be retrieved.
Note: The original project directory created with the `apicup` installer during the initial 2018 product installation (for example, `myProject`) is required if you need to restore the database and to run disaster recovery back to 2018. You cannot restore the database or perform a disaster recovery without the initial project directory because it contains pertinent information about the 2018 cluster.
6. Complete a full backup for the 2018 Management, Portal, and Analytics subsystems.
 - You must back up both the Management and Portal subsystems at the same time, to ensure synchronicity across the services.
 - For the Analytics backup,
 - The S3 repository access/credentials must be set
 - The backup `REPO_NAME` must be `apic-analytics`

Attention: Backups are intended for recovery of the Management, Portal, and Analytics subsystems onto the same deployment from which they were taken, or onto a new replacement installation in the same environment for disaster recovery. The same environment means the same network configuration and project directory as the original installation. Make sure you have current and correct configuration for backup and restore for all of the 2018 subsystems. Verify that you can successfully restore from the backups. For information on backing up and restoring 2018, see [Backing up and restoring](#).

7. Run the following command to create a backup secret for the Analytics subsystem's S3 repository credentials.

```
oc create secret generic analytics-backup-secret -n APIC_namespace \
  --from-literal=access_key='<YOUR-ACCESS-KEY>' \
  --from-literal=secret_key='<YOUR-KEY-SECRET>'
```

The secret must be called `analytics-backup-secret` and must be in the same namespace as the v2018 Analytics deployment. For more information on creating the secret, see [Configuring backup settings for Analytics](#).

This secret is used as part of the backup configuration for the v10 install. While upgrading to version 10.0.5.x, the backup is configured in v10 using this secret; however the analytics data from v2018 is not restored due to changes in the Analytics subsystem (you can export that data in the next step).

8. If you want to retain your analytics data, then you must export it before running the upgrade.

For instructions on exporting data from v2018, see the v2018 documentation: <https://www.ibm.com/docs/en/api-connect/2018.x?topic=analytics-exporting-api-event-data>. Also, you can set up analytics offload so that all new analytics data is sent to a third-party system for storage: <https://www.ibm.com/docs/en/api-connect/2018.x?topic=cloud-configuring-analytics-offload-api-connect>.

9. Disassociate and delete your Analytics services.

Note: If you want your v2018 deployment to continue gathering analytics data up to the last possible moment, this step can instead be done just before the Management subsystem upgrade: [7](#).

- a. In Cloud Manager, click Topology.
- b. In the section for the Availability Zone that contains the Analytics service, locate the Gateway service that the Analytics service is associated with.
- c. Click the actions menu, and select Unassociate analytics service. Remember to disassociate each Analytics service from all Gateways.
- d. In the section for the Availability Zone that contains the Analytics services, locate each Analytics service and click Delete.

10. Limit permissions in the target namespace.

In 2018, it was recommended to apply a broad set of permissions; however, 10.0.5.0 and later use a more limited set of permissions. If your permissions are too broad, the upgrade might fail. For best results, assign a more limited set of permissions now by completing the following steps:

- a. Remove the broad set of permissions by running the following command:

```
oc adm policy remove-scc-from-group anyuid system:serviceaccounts:APIC_namespace
```

- b. List the Helm releases in the 2018 deployment by running the following command:

```
helm ls
```

The response looks like the following example, with the name of each Helm release displaying in the **CHART** column:

NAME	NAMESPACE	REVISION	UPDATED	STATUS	CHART
APP VERSION					
r19b9d4b2eb	e2e	1	2021-02-05 17:22:11.050619495 -0800	PST deployed	apiconnect-2.0.0
r5d63b7fc92	e2e	1	2021-02-05 17:22:24.115569226 -0800	PST deployed	apic-portal-2.0.0
r6c7b1188e0	e2e	1	2021-02-05 17:22:50.169539831 -0800	PST deployed	apic-analytics-2.0.0
r8dacbcffda	e2e	1	2021-02-05 17:22:36.248182901 -0800	PST deployed	dynamic-gateway-service-1.0.63
rff8ba771c8	e2e	1	2021-02-05 17:22:10.329984775 -0800	PST deployed	cassandra-operator-1.0.0

- c. Assign permissions to each of the Helm releases by running the following command:

```
for __sa in $(oc get sa -l release=<helm_release>); do oc adm policy add-scc-to-user anyuid system:serviceaccount:<namespace>:__$sa; done
```

where:

- **<helm_release>** is the name of the Helm release from the **NAME** column; for example **r19b9d4b2eb**
- **APIC_namespace** is the namespace where you installed API Connect.

For example, the sample response lists 5 Helm releases, so you must run the command for each of those names:

- **r19b9d4b2eb**
- **r5d63b7fc92**
- **r6c7b1188e0**
- **r8dacbcffda**
- **rff8ba771c8**

- d. Assign permissions to the default service account by running the following command:

```
oc adm policy add-scc-to-user anyuid system:serviceaccount:APIC_namespace:default
```

11. Upgrade the OpenShift cluster to version 4.10.

- Change the channel in OpenShift, and wait for the upgrade to finish.
- Wait for nodes to all show the newer version of Kubernetes.

12. On 2018, check the health of each subsystem again:

```
apicup subsys health-check <subsystem_name>
```

What to do next

When you are ready to upgrade, proceed to [OpenShift: Upgrading from 2018 to v10](#) for instructions.

Next topic: [OpenShift: Upgrading from 2018 to v10](#)

OpenShift: Upgrading from 2018 to v10

Perform an online upgrade of IBM® API Connect 2018 to the latest release of v10 on OpenShift using a top-level upgrade CR.

Before you begin

Be sure to complete all of the steps in [OpenShift: Preparing the 2018 deployment for upgrade](#). If you need to perform a disaster recovery to revert to 2018, you will need the backed-up **apicup** project as well as the backups of all the API Connect subsystems.

Attention: This task uses a single, top-level upgrade CR to upgrade all API Connect subsystems at once. Only use these instructions if all 4 API Connect v2018 subsystems are installed in a single namespace.

If your v2018 deployment installed subsystems into different namespaces, you must upgrade the subsystems individually using separate upgrade CRs. Follow the instructions for a Kubernetes deployment and upgrade the subsystems in the following sequence:

1. [Upgrading the Management subsystem from 2018 to v10](#)
2. [Upgrading the Developer Portal subsystem from 2018 to v10](#)
3. [Upgrading the Analytics subsystem from v2018 to v10](#)
4. [Upgrading the gateway subsystem from 2018 to v10](#)

About this task

Run the apicops fix on the database, install operators, create the top-level upgrade CR, and then perform the upgrade.

Procedure

1. On the 2018 deployment, use the `apicops` utility to resolve any known inconsistencies and synchronization problems with the database:
The `apicops preupgrade` command must be run in fix mode just prior to starting the management upgrade. The fix mode fixes any inconsistencies and synchronization problems in the database.

Important: Upgrade will fail if the `apicops` command is not successfully completed within 2 hours of starting the Management upgrade. If the Management upgrade must be run more than once, you must run the command in fix mode within 2 hours each time you run the upgrade.

- a. Obtain the latest 2018 release of `apicops` from <https://github.com/ibm-apiconnect/apicops#latest-v2018-release>.
- b. Set up the environment. See <https://github.com/ibm-apiconnect/apicops#requirements>.
- c. Run the `apicops` report with the following command:

```
apicops preupgrade -m report --upgrade_to v10 -l apicops_preupgrade_get_out.txt
```

- d. Run the `apicops` fix with the following command:

```
apicops preupgrade -m fix --upgrade_to v10 -l apicops_preupgrade_fix_out.txt
```

- The `-m` option runs the utility in fix mode. Running the utility in fix mode resolves most problems. If the utility identifies a problem it cannot resolve, contact IBM Support.
- Set the `-l` option to send output to a log file.

If `apicops` indicates that you need to have a `KUBECONFIG` environment variable set then run the following command before trying the fix again:

```
export KUBECONFIG=~/.kube/config
```

2. Take a full backup of the API Connect 2018 Management subsystem. This is in addition to the backup taken in [OpenShift: Preparing the 2018 deployment for upgrade](#), which was taken before the pre-upgrade operations. The pre-upgrade operations will have changed some of the contents of the database and so it is recommended to have backups from before and after.
3. From the OpenShift web console, click + (plus sign), and select create CatalogSource resources.
Create the resources by completing the following steps:

- a. Create the `ioc-catsrc.yaml` file with IBM Operator Catalog source; for example :

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: ibm-operator-catalog
  namespace: openshift-marketplace
spec:
  displayName: "IBM Operator Catalog"
  publisher: IBM
  sourceType: grpc
  image: icr.io/cpopen/ibm-operator-catalog
  updateStrategy:
    registryPoll:
      interval: 45m
```

- b. Apply the IBM Operator Catalog source with the following command:

```
oc apply -f ioc-catsrc.yaml -n openshift-marketplace
```

- c. Create the `cs-catsrc.yaml` file with the IBM Cloud Pak foundational services (previously called Common Services) operator catalog source:

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: opencloud-operators
  namespace: openshift-marketplace
spec:
  displayName: IBMCS Operators
  image: icr.io/cpopen/ibm-common-service-catalog:latest
  publisher: IBM
  sourceType: grpc
  updateStrategy:
    registryPoll:
      interval: 45m
```

- d. Apply the IBM Cloud Pak foundational services operator catalog source with the following command:

```
oc apply -f cs-catsrc.yaml -n openshift-marketplace
```

4. Install the API Connect V2018 Upgrade operator:
 - a. Open the OCP Operator Hub in the namespace where you previously installed API Connect.
 - b. Set the filter to "IBM API Connect V2018 Upgrade" to quickly locate the operators required for API Connect.
 - c. Click the IBM API Connect V2018 Upgrade tile.
 - d. Click Install again and set the following options for the API Connect V2018 Upgrade operator:
 - Set the Update Channel to v2.0.6.
Selecting a channel installs the latest upgrade operator version that is available for the channel.

- For the Install Mode, select A specific namespace on the cluster and select the namespace where you previously installed API Connect
 - Set the Approval Strategy to Automatic
- e. Click Install and wait while the upgrade operator is installed.
 - f. Verify that the operator installed successfully by clicking Installed Operators, selecting the namespace where you installed API Connect, and making sure that the IBM API Connect V2018 Upgrade operator is listed.
5. (10.0.5.2 and earlier) Create a cert-manager Issuer:
- Note: Skip this step if you are upgrading to version 10.0.5.3 or later because the cert-manager Issuer is created automatically.
- Example cert-manager Issuer

```
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "selfsigning-issuer"
  }
spec:
  selfSigned: {}
```

6. Create an entitlement key; for example:

```
oc create secret docker-registry ibm-entitlement-key \
  --docker-username=cp \
  --docker-password=<entitlement_key> \
  --docker-server=cp.icr.io \
  --namespace=<APIC_namespace>
```

7. Create the top-level CR.

- a. In the Operator Hub, click Installed Operators, IBM API Connect V2018 Upgrade
- b. Select the API Connect Upgrade tab.
- c. Create the top-level CR for upgrade by clicking APIConnectUpgradeFromV2018.
- d. Select the YAML view and populate the CR with the settings for your environment.

In particular, provide values for the following settings:

- **license accept** - set to **true** (installation fails if this is left as the default value of **false**)
- **license use** - set to **production** or **nonproduction** according to your license purchase
- Set license: to the License ID for the version of API Connect that you purchased. See [API Connect licenses](#).
- **namespace** - namespace where API Connect is installed
- **storageClassName** - a storage class that is available in your cluster (you must specify a default storage class)
- **certManagerIssuer name** - the name of the cert-manager Issuer that you created in the previous step (or another issuer that you want to use)
- **profile** - the deployment profile that you want to install, see: [API Connect deployment profiles for OpenShift and Cloud Pak for Integration](#).
- **subsystemName** - the name of your V2018 subsystem

Example top-level CR:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APIConnectUpgradeFromV2018
metadata:
  name: topcr-upgrade
  namespace: test1
spec:
  license:
    accept: true
    license: L-GVEN-GFUPVE
    use: nonproduction
  management:
    subsystemName: mgmt
    triggerRollback: false
  gateway:
    subsystemName: gw
    triggerRollback: false
  analytics:
    analyticsSpec:
      storage:
        enabled: true
      shared:
        volumeClaimTemplate:
          storageClassName: rook-ceph-block
          volumeSize: 50Gi
        type: shared
    subsystemName: a7s
    triggerRollback: false
  storageClassName: rook-ceph-block
  profile: n3xc4.m16
  portal:
    adminVolumeClaimTemplate:
      storageClassName: rook-ceph-block
      volumeSize: 12Gi
    backupVolumeClaimTemplate:
      storageClassName: rook-ceph-block
      volumeSize: 30Gi
    databaseLogsVolumeClaimTemplate:
      storageClassName: rook-ceph-block
      volumeSize: 12Gi
    databaseVolumeClaimTemplate:
      storageClassName: rook-ceph-block
      volumeSize: 30Gi
    webVolumeClaimTemplate:
      storageClassName: rook-ceph-block
      volumeSize: 12Gi
    certVolumeClaimTemplate:
```

```

    storageClassName: rook-ceph-block
    volumeSize: 4Gi
    subsystemName: pt1
    triggerRollback: false
    v10InstanceName: topcr

```

Note: Note: To avoid issues with PVCs being sized too small, specify the PVC sizes of your existing 2018 deployment in the `spec.portal` section of the CR. The PVC sizes allocated might need to be adjusted further (beyond the minimum and existing 2018 values) depending on the number and content of the sites required.

The minimum PVC sizes are large enough to cater for a small number of portal sites. As a rough estimate, for every 10 additional small portal sites (with a limited number of apps, apis, consumer orgs, members and portal content) the amount of space allocated should be increased in line with the following estimates:

- `databaseVolumeClaimTemplate`: +10 Gi
- `webVolumeClaimTemplate`: +1 Gi
- `backupVolumeClaimTemplate`: +2 Gi

If you have sites that have a lot of provider org content and lots of documentation, images, and additional content stored in the portal, then you may require a lot more space to be allocated to your PVCs. The following 2 PVCs do not need to be scaled based on the number of sites:

- `databaseLogsVolumeClaimTemplate`
- `adminVolumeClaimTemplate`

e. Click Create to apply the CR and upgrade the deployment.

8. If needed, rollback a failed upgrade and try again:

a. Set `triggerRollback` to `true` in the top-level upgrade CR (in the section under the failed subsystem) to rollback the upgrade for that subsystem.

b. Edit the top-level upgrade CR to correct the issue, and then change the value of `triggerRollback` to `false`, which stats the upgrade of that subsystem.

9. Verify that the upgrade completed.

Keep checking the status using `oc get apic` and wait for the upgrade to complete.

10. Restart all nats-server pods by running the following command:

```
oc -n <namespace> delete po -l app.kubernetes.io/name=natscluster
```

11. Configure backup settings on the upgraded deployment as explained in [Backups on OpenShift](#).

12. Set the `topCRName-mgmt-admin-pass` secret to match the Cloud Manager admin password.

The secret with name `topCRName-mgmt-admin-pass` (containing the Cloud Manager admin password) will have the password that was used when v10 was launched, but this does not impact logging in to Cloud Manager. After the upgrade, complete the following steps to set the secret to match the correct password:

a. Log in to Cloud Manager using the admin password that you used in v2018.

b. Edit the `topCRName-mgmt-admin-pass` secret and set the password to match the v2018 password, so that it is consistent.

What to do next

If the upgrade failed and cannot be corrected, revert to the API Connect 2018 deployment using the disaster recovery process. For information, see [OpenShift: Recovering the V2018 deployment](#).

If the upgrade succeeded, complete the steps in [OpenShift: Post-upgrade steps](#) to remove 2018 files that were left over from the upgrade process.

Previous topic: [OpenShift: Preparing the 2018 deployment for upgrade](#)

Next topic: [OpenShift: Post-upgrade steps](#)

OpenShift: Post-upgrade steps

After you complete the API Connect upgrade, remove files left over from Version 2018 and optionally enable the Automated API behavior testing application.

About this task

Complete the following steps to remove files left over from Version 2018. Optionally, you can enable the Automated API behavior testing application by following the instructions in the last step.

Procedure

1. Enable the Analytics subsystem as explained in [Enabling Analytics after upgrading](#).

2. To ensure that the 10.0.5.x user interface works correctly, clear your browser cache.

3. Delete each Helm release used in 2018:

a. List the Helm releases in the 2018 API Connect deployment by running the following command:

```
helm ls
```

The response looks like the following example, with the name of each Helm release displaying in the `CHART` column:

NAME	NAMESPACE	REVISION	UPDATED	STATUS	CHART
APP VERSION					
r19b9d4b2eb-2.0.0	e2e	1	2021-02-05 17:22:11.050619495 -0800 PST	deployed	apiconnect-
r5d63b7fc92-2.0.0	e2e	1	2021-02-05 17:22:24.115569226 -0800 PST	deployed	apic-portal-
r6c7b1188e0-analytcs-2.0.0	e2e	1	2021-02-05 17:22:50.169539831 -0800 PST	deployed	apic-
r8dacbcffda-gateway-service-1.0.63	e2e	1	2021-02-05 17:22:36.248182901 -0800 PST	deployed	dynamic-
rff8ba771c8-operator-1.0.0	e2e	1	2021-02-05 17:22:10.329984775 -0800 PST	deployed	cassandra-
	1.0.1				

b. For each Helm release, run the following command to delete it:

```
helm uninstall <helm_release_name>
```

where `<helm_release_name>` is the name of the Helm release from the **NAME** column.

For example, in the sample response, there are five Helm releases listed, so you must run the command 5 times.

4. Manually delete any remaining 2018 pods for the current release by running the following commands, where `APIC_namespace` is the namespace where you installed API Connect:

a. `oc delete pod`
`<releaseName>-apim-schema-init-job-xxxx -n`
`<APIC_namespace>`

b. `oc delete pod`
`<releaseName>-lurschema-init-job-xxxx -n`
`<APIC_namespace>`

5. Delete the secrets left over from 2018:

a. Search for `cassop` (cassandra) secrets with the following names:

- `cassandra-backup-auth-secret-xxx`,
- `cassop-client-certs-xxx`,
- `cassop-server-certs-xxx`

b. For each `cassop` secret, run the following command:

```
oc delete secret <Listed_Cassop_Secrets_from_Search> -n <APIC_namespace>
```

where `APIC_namespace` is the namespace where you installed API Connect (if you installed subsystems into different namespaces, specify the namespace for the Management subsystem).

c. Manually delete the following secrets by running each of the specified commands, where `APIC_namespace` is the namespace where you installed API Connect (if you installed subsystems into different namespaces, specify the namespace for the Management subsystem).

- `oc delete secret analytics-proxy-velox-certs-xxx -n`
`<APIC_namespace>`
- `oc delete secret apim-elasticsearch-velox-certs-xxx -n`
`<APIC_namespace>`
- `oc delete secret apim-ui-basic-auth -n`
`<APIC_namespace>`
- `oc delete secret apim-velox-certs-xxx -n`
`<APIC_namespace>`
- `oc delete secret client-downloads-server-velox-certs-xxx -n`
`<APIC_namespace>`
- `oc delete secret db-client-velox-certs-xxx -n`
`<APIC_namespace>`
- `oc delete secret db-velox-certs-xxx -n`
`<APIC_namespace>`
- `oc delete secret juhu-velox-certs-xxx -n`
`<APIC_namespace>`
- `oc delete secret ldap-velox-certs-xxx -n`
`<APIC_namespace>`
- `oc delete secret lur-velox-certs-xxx -n`
`<APIC_namespace>`
- `oc delete secret migration-inputs-xxx -n`
`<APIC_namespace>`
- `oc delete secret migration-velox-certs-xxx -n`
`<APIC_namespace>`
- `oc delete secret plugin-certs-xxx -n`
`<APIC_namespace>`
- `oc delete secret ui-velox-certs-xxx -n`
`<APIC_namespace>`

Attention: When you upgrade to API Connect 10.0.5.x from 2018, some external certificates are carried over from the 2018 deployment and are not managed by cert-manager. When you update any of the retained certificates, you must restart the affected pods manually. For a list of the certificates that are retained after an upgrade and must be updated manually, see [V2018 upgrade: List of certificates to update manually](#), in the [Renewing TLS certificates](#) section of this documentation.

6. Delete any remnant 2018 Analytics cronjob pods.

Two cronjob pods can remain after the upgrade with names matching the format shown below:

- `r*****-analytics-cronjobs-retention-*****-*****`
- `r*****-analytics-cronjobs-rollover-*****-*****`

These are remnant pods from 2018 and are now redundant. To remove them, run the following commands, where `APIC_namespace` is the namespace where you installed API Connect (if you installed subsystems into different namespaces, specify the namespace for the Analytics subsystem).

a. List the pods:

```
oc get pods -n <APIC_namespace>
```

b. Delete the pods:

```
oc delete pods <APIC_namespace> -n <namespace>
```

7. Optional: Configure additional features related to inter-subsystem communication security, such as CA verification and JWT security: [Optional post-upgrade steps for upgrade to 10.0.5.3 from earlier 10.0.5 release](#).

8. Optional: Enable the Automated API behavior testing application.

The Automated API behavior testing provides a simple set of fields where users can quickly invoke APIs to validate them, optionally automating tests and defining schedules for running them. Automated API behavior testing is an optional add-on to API Connect, and must be deployed separately. If you want to install the Automated API behavior testing application, complete the following steps.

- a. Create an ingress-issuer and ingress-ca for use with the Automated API behavior testing application.

Note: The Automated API behavior testing application requires that the API Connect deployment use the cert-manager and the default certificates. Custom certificates are not supported and if used, the Automated API behavior testing application will not install correctly.

The following example shows the configuration of the ingress-issuer and ingress-ca for the Automated API behavior testing application:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ingress-ca
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-ca"
  }
spec:
  duration: 87600h # 10 years
  renewBefore: 720h # 30 days
  secretName: ingress-ca
  commonName: "ingress-ca"
  usages:
  - digital signature
  - key encipherment
  - cert sign
  isCA: true
  issuerRef:
    name: selfsigning-issuer
    kind: Issuer
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-issuer"
  }
spec:
  ca:
    secretName: ingress-ca
---
```

- b. [Add the Automated API behavior testing application to your deployment.](#)

Remember: When you configure the Automated API behavior testing application, make sure to use the same issuer name, `ingress-issuer`, that you created in the previous step.

Previous topic: [OpenShift: Upgrading from 2018 to v10](#)

Next topic: [OpenShift: Recovering the V2018 deployment](#)

OpenShift: Recovering the V2018 deployment

Use the disaster recovery process to revert a failed upgrade to the IBM® API Connect V2018 deployment on OpenShift.

Before you begin

A recovery can only be performed if you saved a backup copy of the apicup project that was created when you installed.

About this task

To revert to V2018, remove artifacts from the failed upgrade, install V2018, and then restore the backups that you created for the V2018 deployment before you started the upgrade.

You can also use this process if you find a need to revert to V2018 after a successful upgrade to V10.

Procedure

1. Delete the upgrade tasks by running the following commands:
 - a. `oc delete up2018g -n <APIC_namespace> --all`
 - b. `oc delete up2018a -n <APIC_namespace> --all`
 - c. `oc delete up2018p -n <APIC_namespace> --all`
 - d. `oc delete up2018m -n <APIC_namespace> --all`where `APIC_namespace` is the namespace where you installed API Connect.
2. Delete all of the V10 subsystems by running the following commands in the sequence shown:
 - a. Delete the Gateway subsystem: `oc delete gw -n <APIC_namespace> --all`
 - b. Delete the Analytics subsystem: `oc delete a7s -n <APIC_namespace> --all`
 - c. Delete the Portal subsystem: `oc delete ptl -n <APIC_namespace> --all`
 - d. Delete the Management subsystem: `oc delete mgmt -n <APIC_namespace> --all`

where `APIC_namespace` is the namespace where you installed API Connect.

Note: To revert to 2018 FP24 and higher, you remove all of the 10.0.5.4 subsystems.

3. Delete the Helm releases that were used in 2018 FP24 and higher.

a. List the Helm releases in the 2018 FP24 and higher API Connect deployment by running the following command:

```
helm ls
```

The response looks like the following example, with the name of each Helm release displaying in the `CHART` column:

NAME	NAMESPACE	REVISION	UPDATED	STATUS	CHART
APP VERSION					
r19b9d4b2eb-2.0.0	e2e	1	2021-02-05 17:22:11.050619495 -0800 PST	deployed	apiconnect-
r5d63b7fc92-2.0.0	e2e	1	2021-02-05 17:22:24.115569226 -0800 PST	deployed	apic-portal-
r6c7b1188e0-analytics-2.0.0	e2e	1	2021-02-05 17:22:50.169539831 -0800 PST	deployed	apic-
r8dachbcffda-gateway-service-1.0.63	e2e	1	2021-02-05 17:22:36.248182901 -0800 PST	deployed	dynamic-
rff8ba771c8-operator-1.0.0	e2e	1	2021-02-05 17:22:10.329984775 -0800 PST	deployed	cassandra-

b. For each Helm release, run the following command to delete it:

```
helm uninstall <helm_release>
```

where `<helm_release>` is the name of the Helm release from the `CHART` column.

For example, in the sample response, there are five Helm releases, so you must run the command to delete each of those releases:

- `apiconnect-2.0.0`
- `apic-portal-2.0.0`
- `apic-analytics-2.0.0`
- `dynamic-gateway-service-1.0.63`
- `cassandra-operator-1.0.0`

4. Delete all PVCs related to the API Connect V2018 and V10 deployments.

• If there is no PVC in the namespace other than those created by API Connect, you can delete them all at once by running the following command:

```
oc delete pvc --all -n <APIC_namespace>
```

• If there are other PVCs in the same namespace, then delete the API Connect PVCs individually by running the following commands:

• Get the list of PVCs in the namespace:

```
oc get pvc
```

• Delete one PVC:

```
oc delete pvc <PVC_name> -n <APIC_namespace>
```

where:

- `PVC_name` is the name of a particular PVC that you want to delete
- `APIC_namespace` is the namespace where you installed API Connect

5. If you do not intend to attempt the V10 upgrade again, use Operator Hub to uninstall the operators and catalog sources related to the upgrade.

6. Install API Connect V2018.

- Use the `apicup` project that you created in V2018 and backed up before starting the upgrade (this is necessary to ensure that settings such as certificates and encryption secrets are retained).
- Grant all of the security permissions required for the V2018 installation.

7. Configure backup and restore settings for each subsystem, and then restore the subsystems with the data you backed up previously in V2018, in the following sequence:

- Management
- Portal
- Analytics

For instructions, see [Backing up and restoring](#) in the V2018 documentation.

Previous topic: [OpenShift: Post-upgrade steps](#)

Optional post-upgrade steps for upgrade to 10.0.5.3 from earlier 10.0.5 release

Version 10.0.5.3 introduces some new features that are related to inter-subsystem communication. This topic explains how to configure these features when you upgrade from a previous 10.0.5 release to v10.0.5.3 or later.

Note: If you are using a top-level CR, you must edit the top-level CR instead of the subsystem CR. If the section you want to edit is missing from the top-level CR, copy it from the subsystem CR and paste it into the top-level CR.

Enabling management CA certification on the REST API

The portal and gateway subsystems make requests to the management subsystem through the REST API that is hosted by the management subsystem. This communication is standard TLS, where each REST API endpoint has a server certificate that is presented to the client during TLS handshaking. For additional security, validation of the management server certificate CA is enabled on all subsystems for new installations of API Connect v10.0.5.3 and later. If you are upgrading from an earlier release, and want to enable this feature for the portal and gateway then edit the portal and gateway CRs, and add the following settings to the `spec`: section depending on if you are using `external` or `in-cluster` inter-subsystem communications ([in-cluster service communication between subsystems](#)):

- If you are using `external` inter-subsystem communication, in the portal CR add:

```
spec:
  mgmtPlatformEndpointCASecret:
    secretName: ingress-ca # Or <instance-name>-ingress-ca if using top-level CR. If you have customized certs, change
    ingress-ca to $PLATFORM_CA_SECRET
  mgmtConsumerEndpointCASecret:
    secretName: ingress-ca # Or <instance-name>-ingress-ca if using top-level CR. If you have customized certs, change
    ingress-ca to $CONSUMER_CA_SECRET
```

and in the gateway CR add:

```
spec:
  mgmtPlatformEndpointCASecret:
    secretName: ingress-ca # Or <instance-name>-ingress-ca if using top-level CR. If you have customized certs, change
    ingress-ca to $PLATFORM_CA_SECRET
```

where:

- `$PLATFORM_CA_SECRET` is the Kubernetes secret object that contains the CA certificate that is used by the platform REST API.
 - `$CONSUMER_CA_SECRET` is the Kubernetes secret object that contains the CA certificate that is used by the consumer REST API.
- If you are using `in-cluster` inter-subsystem communications, in the portal CR add:

```
spec:
  mgmtPlatformEndpointSvcCASecret:
    secretName: management-ca # Or <instance-name>-mgmt-ca if using top-level CR. If you have customized certs, change
    ingress-ca to $PLATFORM_CA_SECRET
  mgmtConsumerEndpointSvcCASecret:
    secretName: management-ca # Or <instance-name>-mgmt-ca if using top-level CR. If you have customized certs, change
    ingress-ca to $CONSUMER_CA_SECRET
```

and in the gateway CR add:

```
spec:
  mgmtPlatformEndpointSvcCASecret:
    secretName: management-ca # Or mgmt-ca or <instance-name>-mgmt-ca if using top-level CR. If you have customized certs,
    change ingress-ca to $PLATFORM_CA_SECRET
```

If changing to use `in-cluster` inter-subsystem communication after an upgrade, you must also reregister all subsystems: [Converting existing registered subsystems to use in-cluster communications](#).

Note: If you are not sure of the secret name, follow these steps:

1. Check the management subsystem CR with `oc describe mgmt`, and identify the issuer of the platform and consumer API endpoints:

```
oc describe mgmt -n <namespace>
...
Platform API Endpoint:
Annotations:
  cert-manager.io/issuer:          apic-ingress-issuer
...
Consumer API Endpoint:
Annotations:
  cert-manager.io/issuer:          apic-ingress-issuer
```

2. Describe the issuer with `oc describe issuer`, and identify the secret name:

```
oc describe issuer apic-ingress-issuer -n <namespace>
...
Spec:
  Ca:
    Secret Name:  apic-ingress-ca
...

```

Enable JWT security for inter-subsystem communication

If you want to use JWT security instead of mTLS for inter-subsystem communication, then follow these steps. For more information on JWT security, see: [Enable JWT security instead of mTLS](#)

Cloud Pak for Integration and OpenShift top-level CR installations

Configure JWT as described in the post-install steps: [Enable JWT security and disable mTLS between subsystems](#).

Individual subsystem CR installations

1. Describe the `mgmt` CR to get the JWKS URL:

```
kubectl describe mgmt -n <namespace>
...
status:
  - name: jwksUrl
    secretName: api-endpoint
    type: API
    uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

2. For each portal, gateway, and analytics subsystem where you want to use JWT security, edit the CR corresponding that subsystem to disable mTLS and specify the `jwksUrl`:

```
spec:
  ...
  mtlsValidateClient: false
  jwksUrl: <JWKS URL>
```

<JWKS URL> is the URL identified in step [1](#).

3. **v10.0.5.4 and later:** Enable JWT on the gateway to analytics communications flow. Enable the Use JWT switch for the registered gateway in the Topology page of the Cloud Manager UI.

Datapower version 10.5.0.4 only: To enable JWT on the gateway to analytics communications flow, you also must add a `dataPowerOverride` section to the gateway CR:

```
spec:
  jwksUrl: <JWKS URL>
  dataPowerOverride:
    image: customregistry.com/custom-image-datapower:10.5.0.5
    version: 10.5.0.5
    license: X-XXXX-XXXXXX
```

Converting existing registered subsystems to use in-cluster communications

If you want to change your existing portal, gateway, and analytics subsystems to use `in-cluster` inter-subsystem communication then they must be reregistered. For more information about `in-cluster` communication, see [In-cluster service communication between subsystems](#).

Important: Understand the following points before you change the communication type of a subsystem:

- Reregistering any of the subsystems requires an outage of that subsystem.
- When you change communication type, any backup of the management subsystem that was taken before this change becomes unusable. Do not change the inter-subsystem communication type if you might want to restore the management subsystem from an earlier backup.
- To reregister the portal, all portal sites must be deleted first. Deleting a portal site means that any customizations that were made to the site must manually be reapplied after the portal is reregistered and the site recreated.
- To reregister the gateway, all published products must be deleted from the gateway.

To reregister a portal subsystem, the steps are:

1. Delete all portal sites from the portal service. From the API Manager UI, for each catalog that has a site on the portal, delete the portal site: [Edit or delete portal site](#).
2. Delete the portal service in the Topology section of the Cloud Manager UI.
3. Reregister the portal service, specifying the inter-subsystem communication type that you want, in the Topology section of the Cloud Manager UI. See [Register portal service](#).

To reregister a gateway subsystem, the steps are:

1. In the API Manager UI, delete all products from all catalogs that use the gateway service. See [Removing a product from a catalog](#).
2. In the API Manager UI, for each catalog that uses the gateway service, remove the gateway service from the catalog settings. See [Configure catalog gateway service](#).
3. Delete the gateway service in the Topology section of the Cloud Manager UI.
4. Reregister the gateway service, specifying the inter-subsystem communication type that you want, in the Topology section of the Cloud Manager UI. See [Registering a gateway service](#).

To reregister the analytics subsystem, the steps are:

1. Unassociate the analytics service from all gateway services that it is associated with. See [Unassociate analytics service](#).
2. Delete the analytics service in the Topology section of the Cloud Manager UI.
3. Reregister the analytics service, specifying the inter-subsystem communication type that you want, in the Topology section of the Cloud Manager UI. See [Registering an analytics service](#).

Update duration of certificates

In earlier releases of API Connect, some of the certificates had a duration of 12720 hours. On new API Connect v10.0.6 installations, all certificates have a duration of 17520 hours. After you upgrade to v10.0.6, update API Connect certificates to have a duration of 17520 hours:

1. List all the certificates that have a 12720 hour duration with the following command:

```
oc -n <namespace> get certificate -o custom-columns=NAME:metadata.name,DURATION:spec.duration | grep 12720h0m0s | awk '{print $1}'
```

Example output:

```
small-a7s-ai-endpoint
small-gw-gateway
small-gw-gateway-manager
small-mgmt-admin
small-mgmt-api-manager
small-mgmt-consumer-api
small-mgmt-platform-api
small-ptl-portal-director
small-ptl-portal-web
```

2. Trigger the recreation of each certificate by deleting the certificate:

```
oc -n <namespace> delete certificate <certificate name>
```

For example:

```
oc -n apic delete certificate small-ptl-portal-web
certificate.cert-manager.io "small-ptl-portal-web" deleted
```

3. Verify that the recreated certificate has a 17520 hour duration:

```
oc -n <namespace> get certificate <certificate name> -o custom-columns=DURATION:spec.duration
```

For example:

```
oc -n apic get certificate small-ptl-portal-web -o custom-columns=DURATION:spec.duration
DURATION
```

Enabling Analytics after upgrading

After the upgrade successfully completes, re-enable Analytics data collection.

Before you begin

If upgrading from v10.0.5 you do not need to do these steps.

About this task

This task is only required if you are upgrading from releases prior to 10.0.5.0.

Procedure

1. Recreate your analytics backup secret:
 - a. Delete the secret: `oc delete secret analytics-backup-secret`
 - b. Re-create the secret with a new name: `oc create secret generic <new backup secret name> --from-literal=access_key='Your_Access_Key' --from-literal=secret_key='Your_access_key_secret' -n Namespace_of_Analytics_Subsystem`
 - c. Update the credentials value in your top-level CR with the new backup secret name. Also, change your backup path to prevent the backups list showing pre-upgrade backups that you cannot restore from. For example:

```
spec:
  analytics:
    databaseBackup:
      ...
      credentials: <new backup secret name>
      path: <new backup path>
      ...
```

If you are not using top-level CR, update these properties in your Analytics CR.



If this step is skipped then backups will not run.

2. Delete the old analytics PVCs. If this step is not done, then your previous (and now inaccessible) analytics data will continue to occupy storage space. List your PVCs and identify those that contain the following strings in their names: `storage-master`, `storage-coord`, `storage-data`, `storage-shared`, `kafka`, `zk`. These PVCs should all have an `AGE` that dates back to before you started your upgrade.

```
kubectl get pvc -n <namespace>
NAME                                STATUS  VOLUME          CAPACITY  ACCESS MODES  STORAGECLASS  AGE
...
data-ger2-analytics-storage-0        Bound   a7s-pv-8735e017  250Gi     RWO            rook-ceph-block  3h29m
data-ger2-analytics-storage-shared-0 Bound   a7s-pv-30aac7ed  250Gi     RWO            rook-ceph-block  227d
...
```

In this example, `data-ger2-analytics-storage-shared-0` contains the string `storage-shared`, and it is 227 days old. To delete it, run:

```
kubectl delete pvc data-ger2-analytics-storage-shared-0 -n <namespace>
```

3. Add the new `analytics:view` permission to the Administrator role in the Cloud Manager UI.
 - a. Click , then select Roles
 - b. Click  on the Administrator role and select Edit
 - c. Select the checkbox next to the analytics permission and click save:

Analytics

View

4. Register new analytics services: [Registering an analytics service](#)
5. Associate your new analytics services with your gateways [Associating an analytics service with a gateway service](#)

Troubleshooting installation and upgrade on OpenShift

Review the following troubleshooting tips if you encounter a problem while installing or upgrading API Connect on OpenShift, including as a component of IBM Cloud Pak for Integration (CP4I).

- [One or more pods in CrashLoopBackoff or Error state, and report a certificate error in the logs](#)
- [PgBouncer pod enters a CrashLoopBackoff state when upgrading to API Connect 10.0.5.3](#)
- [You see the denied: insufficient scope error during an air-gapped deployment](#)
- [Apiconnect operator crashes](#)
- [Upgrading a 3 node profile to IBM API Connect 10.0.3.0 might result in some portal-db/www pods being stuck in the Pending state](#)
- [Issues when installing Drupal 8 based custom modules or sub-themes into the Drupal 9 based Developer Portal](#)
- [Disabling the Portal web endpoint check](#)



Note: In the [Help](#) page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.

One or more pods in `CrashLoopBackoff` or `Error` state, and report a certificate error in the logs

In rare cases, cert-manager might detect a certificate in a bad state right after it has been issued, and then re-issues the certificate. If a CA certificate has been issued twice, the certificate that was signed by the previously issued CA will be left stale and can't be validated by the newly issued CA. In this scenario, one of the following messages displays in the log:

- `javax.net.ssl.SSLHandshakeException: Received fatal alert: certificate_unknown`
- `Error: unable to verify the first certificate`
- `ERROR: openssl verify failed to verify the Portal CA tls.crt, ca.crt chain signed the Portal Server tls.crt cert`

Resolve the problem by completing the following steps:

1. Use `apicops` (v10 version 0.10.57+ required) to validate the certificates in the system:

```
apicops upgrade:stale-certs -n <namespace>
```

2. If any certificate that is managed by cert-manager fails the validation, delete the stale certificate secret:

```
oc delete secret <stale-secret> -n <namespace>
```

Cert-manager automatically generates a new certificate to replace the one you deleted.

3. Use `apicops` to make sure all certificates can be verified successfully:

```
apicops upgrade:stale-certs -n <namespace>
```

Pgbouncer pod enters a `CrashLoopBackoff` state when upgrading to API Connect 10.0.5.3

This issue might be caused by a corrupt compliance entry in the `pgbouncer.ini` file. When the operator is upgraded to v10.0.5.3 but the `ManagementCluster` CR is not yet upgraded, the operator might update the `pgbouncer.ini` file in the `PGBouncer` secret with the older `ManagementCluster` CR's profile file, which does not contain any value for the `compliance pool_size`. As a result, the value gets incorrectly set to the string `<no value>`.

In the majority of cases this bad configuration update is temporary, until the `ManagementCluster` CR's version is also updated as part of the upgrade process and the problem is resolved. The issue will only become evident if the `pgbouncer` pod restarts prior to the `ManagementCluster` CR's version field being updated.

After the operator updates, if the other pods are restarted (specifically the `pgbouncer` pod) before the `ManagementCluster` CR's version is updated, then the `pgbouncer` pod will get stuck in a `CrashLoopBackOff` state due to the missing value for the `compliance pool_size` setting.

The following example shows what the error looks like in the `pgbouncer` logs:

```
Wed May 17 09:00:02 UTC 2023 INFO: Starting pgBouncer..
2023-05-17 09:00:02.897 UTC [24] ERROR syntax error in connection string
2023-05-17 09:00:02.897 UTC [24] ERROR invalid value "host=mgmt-a516d013-postgres port=5432 dbname=compliance pool_size=<no value>" for parameter compliance in configuration (/pgconf/pgbouncer.ini:7)
2023-05-17 09:00:02.897 UTC [24] FATAL cannot load config file
```

Resolve the issue by running the commands in the following script, where `<namespace>` is the API Connect instance's namespace if you deployed using the top-level `APIConnectCluster` CR, or is the management subsystem's namespace if you deploying using individual subsystem CRs.

```
NAMESPACE=<APIC-namespace>
BOUNCER_SECRET=<management-prefix>-postgres-pgbouncer-secret
TEMP_FILE=/tmp/pgbouncer.ini

**Step 1 - Common: Get the existing pgbouncer.ini file**
oc get secret -n $NAMESPACE $BOUNCER_SECRET -o jsonpath='{.data.pgbouncer\.ini}' | base64 -d > $TEMP_FILE

**Step 2 - Linux version: Update the file and use it to patch the Secret on the cluster**
sed 's/<no value>/20/' $TEMP_FILE | base64 -w0 | xargs -I{} kubectl patch secret -n $NAMESPACE $BOUNCER_SECRET --type='json' -p="[{ 'op' : 'replace' , 'path' : '/data/pgbouncer.ini' , 'value' : {} } ]]"

**Step 2 - Mac version: Update the file and use it to patch the Secret on the cluster**
sed 's/<no value>/20/' $TEMP_FILE | base64 -b0 | xargs -S2000 -I{} kubectl patch secret -n $NAMESPACE $BOUNCER_SECRET --type='json' -p="[{ 'op' : 'replace' , 'path' : '/data/pgbouncer.ini' , 'value' : {} } ]]"

# Step 3 - Common: Restart pgbouncer to pick up the updated Secret configuration
oc delete pod <bouncer_pod_name> -n $NAMESPACE
```

You see the denied: `insufficient scope` error during an air-gapped deployment

Problem: You encounter the `denied: insufficient scope` message while mirroring images during an air-gapped installation or upgrade.

Reason: This error occurs when a problem is encountered with the entitlement key used for obtaining images.

Solution: Obtain a new entitlement key by completing the following steps:

1. Log in to the [IBM Container Library](#).
2. In the Container software library, select Get entitlement key.

3. After the Access your container software heading, click Copy key.
4. Copy the key to a safe location.

Apiconnect operator crashes

Problem: During installation (or upgrade), the Apiconnect operator crashes with the following message:

```
panic: unable to build API support: unable to get Group and Resources: unable to retrieve the complete list of server APIs:
packages.operators.coreos.com/v1: the server is currently unable to handle the request
```

```
goroutine 1 [running]:
github.ibm.com/velox/apiconnect-operator/operator-utils/v2/apiversions.GetAPISupport(0x0)
operator-utils/v2/apiversions/api-versions.go:89 +0x1e5
main.main()
ibm-apiconnect/cmd/manager/main.go:188 +0x4ee
```

Additional symptoms:

- Apiconnect operator is in crash loopback status
- Kube apiserver pods log the following information:

```
E1122 18:02:07.853093 18 available_controller.go:437] v1.packages.operators.coreos.com failed with:
failing or missing response from https://10.128.0.3:5443/apis/packages.operators.coreos.com/v1:
bad status from https://10.128.0.3:5443/apis/packages.operators.coreos.com/v1: 401
```

- The IP logged here belongs to the *package server* pod present in the *openshift-operator-lifecycle-manager* namespace
- Package server pods log the following: */apis/packages.operators.coreos.com/v1* API call is being rejected with 401 issue

```
E1122 18:10:25.614179 1 authentication.go:53] Unable to authenticate the request due to an error: x509:
certificate signed by unknown authority I1122 18:10:25.614224 1 httplog.go:90]
verb="GET" URI="/apis/packages.operators.coreos.com/v1" latency=161.243µs resp=401
UserAgent="Go-http-client/2.0" srcIP="10.128.0.1:41370":
```

- Problem is intermittent

Solution:

- If you find the exact symptoms as described, the solution is to delete package server pods in the *openshift-operator-lifecycle-manager* namespace.
- New package server pods will log the **200 Success** message for the same API call.

Upgrading a 3 node profile to IBM API Connect 10.0.3.0 might result in some *portal-db/www* pods being stuck in the *Pending* state

IBM® API Connect 10.0.3.0 introduces the pod anti-affinity required rule, meaning that in a 3 node profile deployment, all 3 db and www pods can run only if there are at least 3 running worker nodes. This rule can cause some upgrades to version 10.0.3.0 to become stuck in the *Pending* state, in which case some extra steps are needed during the upgrade to workaround the issue. See the following example for detailed information about the issue, and how to continue with the upgrade.

Important: You must have a backup of your current deployment before starting the upgrade.

The following example shows 3 worker nodes:

```
$ oc get nodes
NAME                STATUS    ROLES    AGE   VERSION
apimdev0103         Ready     worker   42m   v1.20.0
apimdev0129         Ready     worker   45m   v1.20.0
apimdev1066         Ready     worker   39m   v1.20.0
```

The pods have been scheduled across only 2 of the 3 worker nodes due to a transient problem with *apimdev1066*, as shown in the following pod list. Pods without persistent storage, such as *nginx-X*, can be rescheduled to *apimdev1066* as soon as they are restarted, but any pods with persistent local storage, such as *db-X* and *www-X*, have to be rescheduled onto the same worker node as that is where their files live.

```
$ oc get po -o wide
NAME                READY    STATUS    RESTARTS   AGE    IP                NODE                NOMINATED NODE
READINESS GATES
ejs-portal-nginx-84f57ffd8c-hbf66  1/1     Running   0           5m12s  888.16.109.208   apimdev0103        <none>         <none>
ejs-portal-nginx-84f57ffd8c-mvq96  1/1     Running   0           5m12s  888.16.142.215   apimdev0129        <none>         <none>
ejs-portal-nginx-84f57ffd8c-vpmt1  1/1     Running   0           5m12s  888.16.142.214   apimdev0129        <none>         <none>
ejs-portal-sitel-db-0              2/2     Running   0           4m39s  888.16.109.209   apimdev0103        <none>         <none>
ejs-portal-sitel-db-1              2/2     Running   0           6m37s  888.16.109.206   apimdev0103        <none>         <none>
ejs-portal-sitel-db-2              2/2     Running   0           4m39s  888.16.142.216   apimdev0129        <none>         <none>
ejs-portal-sitel-www-0             2/2     Running   0           4m9s   888.16.109.210   apimdev0103        <none>         <none>
ejs-portal-sitel-www-1             2/2     Running   0           6m37s  888.16.142.213   apimdev0129        <none>         <none>
ejs-portal-sitel-www-2             2/2     Running   0           4m9s   888.16.142.217   apimdev0129        <none>         <none>
ibm-apiconnect-75b47f9f87-p25dd    1/1     Running   0           5m12s  888.16.109.207   apimdev0103        <none>         <none>
```

The upgrade to version 10.0.3.0 is started and the following pod list is observed:

```
$ oc get po -o wide
NAME                READY    STATUS    RESTARTS   AGE    IP                NODE                NOMINATED NODE
READINESS GATES
ejs-portal-nginx-84f57ffd8c-hbf66  1/1     Running   0           10m    888.16.109.208   apimdev0103        <none>         <none>
ejs-portal-nginx-84f57ffd8c-mvq96  1/1     Running   0           10m    888.16.142.215   apimdev0129        <none>         <none>
ejs-portal-nginx-84f57ffd8c-vpmt1  1/1     Running   0           10m    888.16.142.214   apimdev0129        <none>         <none>
ejs-portal-sitel-db-0              2/2     Running   0           10m    888.16.109.209   apimdev0103        <none>         <none>
ejs-portal-sitel-db-1              0/2     Pending   0           91s    <none>           <none>           <none>         <none>
ejs-portal-sitel-db-2              2/2     Running   0           2m41s  888.16.142.218   apimdev0129        <none>         <none>
ejs-portal-sitel-www-0             2/2     Running   0           9m51s  888.16.109.210   apimdev0103        <none>         <none>
ejs-portal-sitel-www-1             2/2     Running   0           12m    888.16.142.213   apimdev0129        <none>         <none>
ejs-portal-sitel-www-2             0/2     Pending   0           111s   <none>           <none>           <none>         <none>
ibm-apiconnect-75b47f9f87-p25dd    1/1     Running   0           10m    888.16.109.207   apimdev0103        <none>         <none>
```


The pod list shows that `db-2` has restarted, and has been rescheduled to `apimdev0129` as there were no other db pods running on that node. However, `db-1` and `www-2` are both stuck in `Pending` state, as there is already a pod of the same type running on the worker node that is hosting the local storage that they are bound to. If you run a `describe` command of either pod you will see the following output:

Events:

```

Type          Reason          Age           From          Message
----          -
Warning       FailedScheduling 10s (x4 over 2m59s) default-scheduler 0/3 nodes are available: 1 node(s) didn't match pod affinity/anti-affinity, 1 node(s) didn't match pod anti-affinity rules, 2 node(s) had volume node affinity conflict.

```

To resolve this situation you need to delete the PVCs for each pod, and then delete the pod itself, so that Kubernetes will regenerate the PVCs and schedule the pod on the worker node that does not have the anti-affinity conflict.

Therefore, for the `db-1` pod the following commands must be run:

```

$ oc get pvc | grep ejs-portal-sitel-db-1
db-ejs-portal-sitel-db-1          Bound          local-pv-fa445e30    250Gi        RWO          local-storage    15m
dblogs-ejs-portal-sitel-db-1     Bound          local-pv-d57910e7   250Gi        RWO          local-storage    15m

```

```

$ oc delete pvc db-ejs-portal-sitel-db-1 dblogs-ejs-portal-sitel-db-1
persistentvolumeclaim "db-ejs-portal-sitel-db-1" deleted
persistentvolumeclaim "dblogs-ejs-portal-sitel-db-1" deleted

```

```

$ oc delete po ejs-portal-sitel-db-1
pod "ejs-portal-sitel-db-1" deleted

```

For the `www-2` pod the following commands must be run:

```

$ oc get pvc | grep ejs-portal-sitel-www-2
admin-ejs-portal-sitel-www-2     Bound          local-pv-48799536   245Gi        RWO          local-storage    51m
backup-ejs-portal-sitel-www-2    Bound          local-pv-a93f5607   245Gi        RWO          local-storage    51m
web-ejs-portal-sitel-www-2       Bound          local-pv-facd4489   245Gi        RWO          local-storage    51m

```

```

$ oc delete pvc admin-ejs-portal-sitel-www-2 backup-ejs-portal-sitel-www-2 web-ejs-portal-sitel-www-2
persistentvolumeclaim "admin-ejs-portal-sitel-www-2" deleted
persistentvolumeclaim "backup-ejs-portal-sitel-www-2" deleted
persistentvolumeclaim "web-ejs-portal-sitel-www-2" deleted

```

```

$ oc delete po ejs-portal-sitel-www-2
pod "ejs-portal-sitel-www-2" deleted

```

If the PVC has `persistentVolumeReclaimPolicy: Delete` set on it, no cleanup is necessary, as the old data will have been deleted on the worker node that is no longer running the `db-1` and `www-2` pods. However, if you are using local-storage in a Kubernetes installation, then there might be some cleanup to do to remove the old data from the worker node that had the anti-affinity conflict.

Kubernetes can now reschedule the pods. All pods with persistent storage are now spread across the available worker nodes, and the pods whose PVCs were deleted will get a full copy of the data from the existing running pods. The following pod list is now observed in our example:

```

$ oc get po -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE                                NOMINATED NODE
READINESS GATES
ejs-portal-nginx-84f57ffd8c-f85wm    1/1     Running   0           30s   888.16.29.136  apimdev1066                       <none>
ejs-portal-nginx-84f57ffd8c-k5klb    1/1     Running   0           103s  888.16.142.220 apimdev0129                       <none>
ejs-portal-nginx-84f57ffd8c-lqhqs    1/1     Running   0           1m53s 888.16.109.212 apimdev0103                       <none>
ejs-portal-sitel-db-0                2/2     Running   0           6m43s 888.16.109.211 apimdev0103                       <none>
ejs-portal-sitel-db-1                2/2     Running   0           8m20s 888.16.29.134  apimdev1066                       <none>
ejs-portal-sitel-db-2                2/2     Running   0           14m   888.16.142.218 apimdev0129                       <none>
ejs-portal-sitel-www-0               2/2     Running   0           93s   888.16.109.213 apimdev0103                       <none>
ejs-portal-sitel-www-1               2/2     Running   0           3m55s 888.16.142.219 apimdev0129                       <none>
ejs-portal-sitel-www-2               2/2     Running   0           7m27s 888.16.29.135  apimdev1066                       <none>
ibm-apiconnect-75b47f9f87-p25dd     1/1     Running   0           22m   888.16.109.207 apimdev0103                       <none>

```

Issues when installing Drupal 8 based custom modules or sub-themes into the Drupal 9 based Developer Portal

From IBM API Connect 10.0.3.0, the Developer Portal is based on the Drupal 9 content management system. If you want to install Drupal 8 custom modules or sub-themes into the Drupal 9 based Developer Portal, you must ensure that they are compatible with Drupal 9, including any custom code that they contain, and not using any deprecated APIs, for example. There are tools available for checking your custom code, such as [Drupal check](#) on GitHub, which checks Drupal code for deprecations.

For example, any Developer Portal sites that contain modules or sub-themes that don't contain a Drupal 9 version declaration will fail to upgrade, and errors like the following output will be seen in the `admin` logs:

```

[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: Checking theme: emeraldgreen
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '' found for emeraldgreen
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: Checking theme: rubyred
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '8.x' found for rubyred
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: Found themes incompatible with
Drupal 9: emeraldgreen rubyred
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: /tmp/restore_site.355ec8 is NOT
Drupal 9 compatible
...
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: Checking module: custom_mod_1
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '' found for custom_mod_1
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: Checking module: custom_mod_2
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '8.x' found for custom_mod_2
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: Found modules incompatible with
Drupal 9: emeraldgreen rubyred

```

```
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: site1.com is NOT Drupal 9 compatible
```

To fix version compatibility errors, all custom modules and sub-themes should declare a `core_version_requirement` key in their `*.info.yml` file that indicates Drupal 9 compatibility. For example:

```
name: Example module
type: module
description: Purely an example
core: 8.x
core_version_requirement: '^8 || ^9'
package: Example module
```

```
# Information added by Drupal.org packaging script on 2020-05-31
version: '8.x-1.3'
project: 'example_module'
datestamp: 1590905415
```

This example specifies that the module is compatible with all versions of Drupal 8 and 9. For more information, see [Let Drupal know about your module with an .info.yml file](#) on the drupal.org website.

If you have a backup of a site that you need to restore, and are getting the version compatibility error, but the module or theme `*.info.yml` file cannot be changed easily, then you have two options. Either:

- Add an environment variable into the portal CR for the `www` pod of the `admin` container stating `SKIP_D9_COMPAT_CHECK: "true"`. However, if you choose this method, you must be positive that all of the custom modules and themes for your sites are Drupal 9 compatible, as otherwise the sites may end up inaccessible after the upgrade or restore. On OpenShift or IBM Cloud Pak for Integration, edit the top level CR by using the UI, and add the environment variable as shown in the following CR snippet:

```
spec:
  ...
  portal:
    template:
      - containers:
        - env:
          - name: SKIP_D9_COMPAT_CHECK
            value: "true"
          name: admin
        name: www
```

Or:

- Extract the site backup, edit the relevant files inside it, and then tar the backup file again. Note that this procedure will overwrite the original backup file, so ensure that you keep a separate copy of the original file before you start the extraction. For example:
 1. `mkdir /tmp/backup`
 2. `cd /tmp/backup`
 3. `tar xzf path_to_backup.tar.gz`
 4. Edit the custom module and theme files to make them Drupal 9 compatible, and add the correct `core_version_requirement` setting.
 5. `rm -f path_to_backup.tar.gz`
 6. `tar cfz path_to_backup.tar.gz`
 7. `cd /`
 8. `rm -rf /tmp/backup`

Disabling the Portal web endpoint check

When you create or register a Developer Portal service, the Portal subsystem checks that the Portal web endpoint is accessible. However sometimes, for example due to the complexity of public and private networks, the endpoint cannot be reached. The following example shows the errors that you might see in the `portal-www` pod, admin container logs, if the endpoint cannot be reached:

```
An error occurred contacting the provided portal web endpoint: example.com
The provided Portal web endpoint example.com returned HTTP status code 504
```

In this instance, you can disable the Portal web endpoint check so that the Developer Portal service can be created successfully. To disable the endpoint check, complete the following update:

On Kubernetes, OpenShift, and IBM Cloud Pak for Integration
Add the following section to the Portal custom resource (CR) template:

```
spec:
  template:
    - containers:
      - env:
        - name: PORTAL_SKIP_WEB_ENDPOINT_VALIDATION
          value: "true"
        name: admin
      name: www
```

Upgrading a two data center deployment on Kubernetes and OpenShift

How to upgrade a two data center disaster recovery (DR) deployment on Kubernetes and OpenShift.

To upgrade API Connect in a 2DCDR deployment, use the upgrade instructions for your platform, but with the extra considerations and steps documented in this topic.

- [Upgrading on Kubernetes.](#)
- [Upgrading on OpenShift and Cloud Pak for Integration.](#)

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.

Before you begin

- Ensure that API Connect is running in both data centers and that replication is working: [Verifying replication between data centers](#).
- Ensure that you have recent backups of Management and Portal subsystems: [Backup and restore requirements for a two data center deployment](#).

Key points applicable to both Management and Portal subsystems

- Your API Connect deployments must be upgraded to the same API Connect release, down to the interim fix level.
- Both data centers must be upgraded in the same maintenance window.
- Upgrade the data center that has the `ingress-ca` Kubernetes cert-manager certificate object first.
- Extra steps must be taken to ensure that the `ingress-ca` secret in both data centers contains the same `ingress-ca` X.509 certificate

The X.509 CA certificate that is used for TLS communication between the API Connect deployments in each data center is referred to as the `ingress-ca` X.509 certificate. This certificate is stored in the `ingress-ca` secret in both data centers, and it must be the same in each data center for 2DCDR to function. The cert-manager has a separate object that is referred to as the `ingress-ca` Kubernetes cert-manager certificate object. This object exists on the data center that was the original active data center on installation.

Key points:

- The `ingress-ca` Kubernetes cert-manager certificate object exists on only one of the data centers in a 2DCDR deployment.
- The `ingress-ca` X.509 certificate is contained in the `ingress-ca` secret of both data centers in a 2DCDR deployment, as shown:

```
kubectl get secret ingress-ca -n <namespace> -o yaml

apiVersion: v1
data:
  ca.crt: <long cert string>
  tls.crt: <long cert string>
  tls.key: <long cert string>
kind: Secret
...
```

To identify the data center that has the `ingress-ca` Kubernetes cert-manager certificate object, run:

```
kubectl -n <namespace> get cert ingress-ca
```

NAME	READY	SECRET	AGE
ingress-ca	True	ingress-ca	30h

Note: If the data center that has the `ingress-ca` Kubernetes cert-manager certificate object is unrecoverable, a new `ingress-ca` Kubernetes cert-manager certificate object can be created on the active data center by using the `ingress-ca` certificate section of the installation yml file: `helper_files/ingress-issuer-v1.yml`:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ingress-ca
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-ca"
  }
spec:
  secretName: ingress-ca
  commonName: "ingress-ca"
  usages:
  - digital signature
  - key encipherment
  - cert sign
  isCA: true
  duration: 87600h # 10 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  issuerRef:
    name: selfsigning-issuer
    kind: Issuer
```

Pre-upgrade operations in the upgrade documentation for your platform might result in an update to the `ingress-ca` X.509 certificate. Extra steps must then be taken during the 2DCDR deployment upgrade to ensure that both data centers always use the same `ingress-ca` X.509 certificate.

Note: If you are not using cert-manager and you customized your certificates, the `ingress-ca` certificate might have a different name. Ensure that the CA certificate that is used by both data centers is the same during all stages of the upgrade.

Steps for Management upgrades

1. Remove the `multiSiteHA` section from the management CRs in both data centers, converting them both to stand-alone deployments.
2. Verify that the management subsystem in both data centers is running as a stand-alone, and that all the pods are running:

```
kubectl get mgmt -n <namespace>
```

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
<management instance name>	15/15	Running	<version>	<version>-xxxx		14d

If one of the data centers returns 6/6 pods then this means it is still running as a warm-standby, wait a few minutes for it to complete the transition to a stand-alone data center. Run `kubectl`

describe and confirm that the `multiSiteHA` section is gone from the `spec` section, and that the `Status` section shows `ha mode` of active in both data centers:

```
kubectl describe mgmt -n <namespace>
...
Status
  Ha Mode:          active
...
```

3. Upgrade both data centers.

4. Verify that both data centers have the same `ingress-ca` X.509 certificate in their `ingress-ca` secret. Run the following command in both data centers and check that the output is the same:

```
openssl x509 -noout -fingerprint -sha256 -in <(kubectl get secret ingress-ca -n <namespace> -o yaml | grep "^ tls.crt:" | awk '{print $2}' | base64 -d)
```

if you do not have the `openssl` command available, you can instead run only the `kubectl` part, which produces a larger output:

```
kubectl get secret ingress-ca -n <namespace> -o yaml | grep "^ tls.crt:" | awk '{print $2}' | base64 -d
```

if the outputs are different, follow these steps to synchronize the certificates: [Synchronizing ingress-ca X.509 certificates](#).

5. Add the `multiSiteHA` sections to the management CRs, setting one of them to be the active and the other to be the warm-standby.

Steps for Portal upgrades

1. Verify that both data centers have the same `ingress-ca` X.509 certificate in their `ingress-ca` secret. Run the following command in both data centers and check that the output is the same:

```
openssl x509 -noout -fingerprint -sha256 -in <(kubectl get secret ingress-ca -n <namespace> -o yaml | grep "^ tls.crt:" | awk '{print $2}' | base64 -d)
```

if you do not have the `openssl` command available, you can instead run only the `kubectl` part, which produces a larger output:

```
kubectl get secret ingress-ca -n <namespace> -o yaml | grep "^ tls.crt:" | awk '{print $2}' | base64 -d
```

if the outputs are different, follow these steps to synchronize the certificates: [Synchronizing ingress-ca X.509 certificates](#).

2. Determine which data center has the `ingress-ca` Kubernetes cert-manager certificate object:

```
kubectl get certificates -n <portal namespace> | grep ingress-ca
```

If this data center is the active, failover the portal subsystem to make it the warm-standby data center: [How to failover API Connect from the active to the warm-standby data center](#).

3. Start the upgrade of your warm-standby data center by following the upgrade documentation for your platform. Stop at the point where the portal CR is updated with the new API Connect version.

4. Verify that both data centers still have the same `ingress-ca` X.509 certificate, repeating step 1. If they are different, then follow these steps: [Synchronizing ingress-ca X.509 certificates](#).

5. Complete the upgrade of your warm-standby data center by updating the portal subsystem CR, following the remaining upgrade steps for your platform.

6. Start the upgrade of your active data center by following the upgrade documentation for your platform. Stop at the point where the portal CR is updated with the new API Connect version.

7. Verify that both data centers still have the same `ingress-ca` X.509 certificate, repeating step 1. If they are different, then follow these steps: [Synchronizing ingress-ca X.509 certificates](#).

8. Upgrade the portal subsystem in your active data center by updating the portal subsystem CR, following the remaining upgrade steps for your platform.

Synchronizing the ingress-ca X.509 certificate across data centers

Follow these steps to extract your `ingress-ca` X.509 certificate from your source data center and prepare it for application on your target data center:

1. Determine which data center has the `ingress-ca` Kubernetes cert-manager certificate object certificate:

```
kubectl get certificates -n <namespace> | grep ingress-ca
```

this is your source data center.

2. Extract the `ingress-ca` secret from your source data center to a file called `new-ca-issuer-secret.yaml`:

```
kubectl get secret ingress-ca -o yaml -n <namespace> > new-ca-issuer-secret.yaml
```

3. Edit the `new-ca-issuer-secret.yaml` file and remove the `creationTimestamp`, `resourceVersion`, `uid`, `namespace`, and `managedFields`. Remove the labels and annotations sections completely. The resulting contents should include the `ingress-ca` X.509 certificate, and the secret name:

```
apiVersion: v1
data:
  ca.crt: <long cert string>
  tls.crt: <long cert string>
  tls.key: <long cert string>
kind: Secret
metadata:
  name: ingress-ca
  type: kubernetes.io/tls
```

4. Copy the `new-ca-issuer-secret.yaml` to the target data center.

Follow these steps to apply the extracted `ingress-ca` X.509 certificate on your target data center:

1. To apply the `new-ca-issuer-secret.yaml` file, run:

```
kubectl apply -f new-ca-issuer-secret.yaml -n <namespace>
```

2. Regenerate all `ingress-ca` end-entity certificates:

```
kubectl get secrets -n <namespace> -o custom-columns='NAME:.metadata.name,ISSUER:.metadata.annotations.cert-manager.io/issuer-name' --no-headers=true | grep ingress-issuer | awk '{ print $1 }' | xargs kubectl delete secret -n <namespace>
```

All affected pods should automatically restart. For more information on regenerating certificates, see: [Renewing certificates with cert-manager](#).

How to upgrade when one data center is down

If API Connect is still running on the failed data center, follow the steps that are documented previously to upgrade both data centers, before you bring the failed data center back online.

If the failed data center is expected to be down for a long time, you can convert the active data center to a stand-alone data center following these steps: [Removing a two data center deployment](#), but note the following points:

1. Ensure that the network links to the failed data center are removed.
2. Ensure that the failed data center is set to warm-standby in the **multiSiteHA** section. Do not proceed to the next step until the data center completes the transition to warm-standby. View the status of the management and portal CRs to confirm that **HA Mode** reports "passive".
3. Remove the **multiSiteHA** section from failed data center, and ensure that the failed data center resets itself to become an empty stand-alone API Connect deployment (all data is deleted).
4. Before you restore the network links between the data centers, do the following:
 - Upgrade API Connect on the failed data center to the same version as the active.
 - Add the **multiSiteHA** sections to both data centers, setting the failed data center to be warm-standby.Important: Do not set the failed data center to be active in the **multiSiteHA** section because it results in an overwrite of the data on your working data center with the empty database of your failed data center.

Related concepts

- [Two data center deployment strategy on Kubernetes and OpenShift](#)

Related tasks

- [Installing a two data center deployment on Kubernetes](#)

Upgrading a two data center deployment on Cloud Pak for Integration

How to upgrade a two data center disaster recovery (DR) deployment on Cloud Pak for Integration.

To upgrade a Cloud Pak for Integration deployment of API Connect in a 2DCDR deployment, use the Cloud Pak for Integration upgrade instructions [Upgrading API Connect in Cloud Pak for Integration](#), but with the extra considerations and steps documented in this topic.

Before you begin

- Ensure that API Connect is running in both data centers and that replication is working: [Verifying replication between data centers](#).
- Ensure that you have recent backups of Management and Portal subsystems: [Backup and restore requirements for a two data center deployment](#).

Key points applicable to both Management and Portal subsystems

- Your API Connect deployments must be upgraded to the same API Connect release, down to the interim fix level.
- Both data centers must be upgraded in the same maintenance window.
- Extra steps must be taken to ensure that the **ingress-ca** secret in both data centers contains the same **ingress-ca** X.509 certificate

The X.509 CA certificate that is used for TLS communication between the API Connect deployments in each data center is referred to as the **ingress-ca** X.509 certificate. This certificate is stored in the **ingress-ca** secret in both data centers, and it must be the same in each data center for 2DCDR to function. The cert-manager has a separate object that is referred to as the **ingress-ca** Kubernetes cert-manager certificate object.

Pre-upgrade operations might result in an update to the **ingress-ca** X.509 certificate. Extra steps must then be taken during the 2DCDR deployment upgrade to ensure that both data centers always use the same **ingress-ca** X.509 certificate.

2DCDR upgrade steps for Cloud Pak for Integration

In Cloud Pak for Integration, the Management and Portal subsystems are managed by the **apiconnectcluster** CR (also known as the top-level CR). When the **apiconnectcluster** CR is updated with the new API Connect version, all the subsystems are updated automatically. Follow the upgrade process for Cloud Pak for Integration, starting from [Upgrading API Connect in Cloud Pak for Integration](#), along with the following steps.

1. Verify that both data centers have the same **ingress-ca** X.509 certificate in their **<apic instance-name>-ingress-ca** secret. Run the following command in both data centers and check that the output is the same:

```
openssl x509 -noout -fingerprint -sha256 -in <(oc get secret <apic instance-name>-ingress-ca -n <namespace> -o yaml | grep '^  tls.crt:' | awk '{print $2}' | base64 -d)
```

if you do not have the **openssl** command available, you can instead run only the **oc** part, which produces a larger output:

```
oc get secret <apic instance-name>-ingress-ca -n <namespace> -o yaml | grep '^  tls.crt:' | awk '{print $2}' | base64 -d
```

if the outputs are different, follow these steps to synchronize the certificates: [Synchronizing the ingress-ca X.509 certificate across data centers](#).

2. Remove the **multiSiteHA** section from the Management sections of the **apiconnectcluster** CRs in both data centers, converting them both to stand-alone management deployments. Be careful to do this only in the management sections, and not the portal sections of the **apiconnectcluster** CR.

3. Start the upgrade of your portal warm-standby data center by following the upgrade documentation for the API Management capability in <https://www.ibm.com/docs/en/cloud-paks/cp-integration>. Stop at the point after the API Connect and DataPower operators are updated but before the operands are updated.
4. Repeat step 1 to verify that the `ingress-ca` X.509 certificates are still the same in both data centers.
5. Complete the upgrade of your portal warm-standby data center, updating the operands as described in the Cloud Pak for Integration documentation.
6. Repeat step 1 to verify that the `ingress-ca` X.509 certificates are still the same in both data centers.
7. Start the upgrade of your portal active data center by following the upgrade documentation for the API Management capability in <https://www.ibm.com/docs/en/cloud-paks/cp-integration>. Stop at the point after the API Connect and DataPower operators are updated but before the operands are updated.
8. Repeat step 1 to verify that the `ingress-ca` X.509 certificates are still the same in both data centers.
9. Complete the upgrade of the portal active data center, updating the operands as described in the Cloud Pak for Integration documentation.
10. Add the `multiSiteHA` sections to the management section of the `apiconnectcluster` CRs in both data centers. Set the same active data center for management as for portal.

Synchronizing the `ingress-ca` X.509 certificate across data centers

Follow these steps to extract your `ingress-ca` X.509 certificate from your source data center and prepare it for application on your target data center. Your source data center is the data center that you are upgrading first:

1. Extract the `<apic instance-name>-ingress-ca` secret from your source data center to a file called `new-ca-issuer-secret.yaml`:

```
oc get secret <apic instance-name>-ingress-ca -o yaml -n <namespace> > new-ca-issuer-secret.yaml
```

2. Edit the `new-ca-issuer-secret.yaml` file and remove the `creationTimestamp`, `resourceVersion`, `uid`, `namespace`, and `managedFields`. Remove the labels and annotations sections completely. The resulting contents should include the `ingress-ca` X.509 certificate, and the secret name:

```
apiVersion: v1
data:
  ca.crt: <long cert string>
  tls.crt: <long cert string>
  tls.key: <long cert string>
kind: Secret
metadata:
  name: ingress-ca
  type: kubernetes.io/tls
```

3. Copy the `new-ca-issuer-secret.yaml` to the target data center.

Follow these steps to apply the extracted `ingress-ca` X.509 certificate on your target data center:

1. To apply the `new-ca-issuer-secret.yaml` file, run:

```
oc apply -f new-ca-issuer-secret.yaml -n <namespace>
```

2. Regenerate all `ingress-ca` end-entity certificates:

```
oc get secrets -n <namespace> -o custom-columns='NAME:.metadata.name,ISSUER:.metadata.annotations.cert-manager.io/issuer-name' --no-headers=true | grep ingress-issuer | awk '{ print $1 }' | xargs oc delete secret -n <namespace>
```

All affected pods should automatically restart. For more information on regenerating certificates, see: [Renewing certificates with cert-manager](#).

How to upgrade when one data center is down

If API Connect is still running on the failed data center, follow the steps that are documented previously to upgrade both data centers, before you bring the failed data center back online.

If the failed data center is expected to be down for a long time, you can convert the active data center to a stand-alone data center following these steps: [Removing a two data center deployment](#), but note the following points:

1. Ensure that the network links to the failed data center are removed.
2. Ensure that the failed data center is set to warm-standby in the `multiSiteHA` section. Do not proceed to the next step until the data center completes the transition to warm-standby. View the status of the management and portal CRs to confirm that `HA Mode` reports "passive".
3. Remove the `multiSiteHA` section from failed data center, and ensure that the failed data center resets itself to become an empty stand-alone API Connect deployment (all data is deleted).
4. Before you restore the network links between the data centers, do the following:
 - Upgrade API Connect on the failed data center to the same version as the active.
 - Add the `multiSiteHA` sections to both data centers, setting the failed data center to be warm-standby.

Important: Do not set the failed data center to be active in the `multiSiteHA` section because it results in an overwrite of the data on your working data center with the empty database of your failed data center.

Maintaining API Connect

You can use utilities to complete maintenance tasks such as backup, restore, and certificate management on Kubernetes.

Note: When maintaining API Connect, do not use `kubectl exec` or `oc exec` commands to access API Connect pods unless advised by IBM.

- [Advanced configuration](#)
You can optionally install and configure advanced features.
- [Backing up and restoring](#)
You can backup and restore API Connect subsystems.

- [Disaster recovery](#)
Prepare for, or recover from, a disaster affecting an API Connect deployment running on Kubernetes, OpenShift, or Cloud Pak for Integration.
- [Disabling and re-enabling the Analytics subsystem on Kubernetes](#)
During maintenance of an API Connect cluster, you can shut down the Analytics subsystem by disabling the ingestion, and storage microservices so that those processes are not running and there is no way to reach them.
- [Disabling and re-enabling the Analytics subsystem on OpenShift and Cloud Pak for Integration](#)
During maintenance of an API Connect cluster, you can shut down the Analytics subsystem by disabling the client, ingestion, and storage microservices so that those processes are not running and there is no way to reach them.
- [Maintaining a two data center deployment](#)
How to maintain a two data center disaster recovery (DR) deployment on Kubernetes and OpenShift, including information about normal operation, failure handling, and recovery.
- [Monitoring and health checks](#)
Monitor your deployment and complete health checks on subsystems.
- [Logging](#)
Configure logging for your deployment on Kubernetes.
- [Troubleshooting](#)
Troubleshoot API Connect subsystems on Kubernetes.

Advanced configuration

You can optionally install and configure advanced features.

See:

- [Installing License Service](#)
License Service helps you assess whether your IBM® API Connect deployment in a Kubernetes environment is compliant with licensing requirements.
- [Installing the Automated API behavior testing application](#)
Update the custom resource for installing IBM API Connect to deploy the Automated API behavior testing application on Kubernetes.
- [Installing the toolkit](#)
- [Changing deployment profiles on Kubernetes](#)
You can change an API Connect installation to use different deployment profiles.
- [Changing deployment profiles on OpenShift](#)
You can change an API Connect installation to use different profiles.
- [Replica override for microservices](#)
Modify the replica count for microservices by using template overrides for Custom Resources.
- [Changing the deployment profile on Cloud Pak for Integration](#)
You can change an API Connect installation to use a different profile.
- [Enabling Developer Portal feature flags on Kubernetes](#)
Use a template override to update the installation CR with settings that control the default behavior of the Developer Portal subsystem.
- [Enabling Developer Portal feature flags on OpenShift and Cloud Pak for Integration](#)
Use a template override to update the installation CR with settings that control the default behavior of the Developer Portal subsystem.
- [Enabling UI feature flags on Kubernetes](#)
Use a template override to update the installation CR with settings that control UI features.
- [Enabling UI feature flags on OpenShift and Cloud Pak for Integration](#)
Use a template override to update the installation CR with settings that control UI features.
- [Advanced configuration for the management subsystem](#)
Specify advanced configuration for the management subsystem.
- [Advanced configuration for the analytics subsystem](#)
Specify advanced configuration for the analytics subsystem on Kubernetes, OpenShift, or Cloud Pak for Integration.
- [Advanced configuration for the gateway subsystem](#)
Specify advanced configuration for the gateway subsystem.

Installing License Service

License Service helps you assess whether your IBM® API Connect deployment in a Kubernetes environment is compliant with licensing requirements.

About this task

License Service provides useful features for managing virtualized environments and measuring license utilization. License Service discovers the software that is installed in your infrastructure, helps you to analyze the consumption data, and allows you to generate audit reports. Each report provides you with different information about your infrastructure, for example the computer groups, software installations, and the content of your software catalog.

By default, every License Service audit report presents data from the previous 90 days. You can customize the type and amount of information that is displayed in a report by using filters, and save your personal settings for future use. You can also export the reports to .csv or .pdf format, and schedule report emails so that specified recipients are notified when important events occur.

When you install API Connect, the `license-use` setting that is configured in the `ibm-apiconnect` operator CR (custom resource) for a particular subsystem is reflected in the Pod annotations. For the Analytics, Portal, and Management subsystems the `license-use` value can be "production" or "nonproduction". For the Gateway subsystem, the `license-use` value can be "production", "nonproduction", or "developer". The subsystem pod annotations are scanned by the `ibm-licensing-operator`, which reports license and CPU usage for each subsystem.

For more information, see [License Service](#) in the IBM Cloud Pak foundational services documentation.

Procedure

Install License Service as explained in the installation instructions in [License Service](#) in the IBM Cloud Pak foundational services documentation. The instructions describe the appropriate License Service installation procedure for API Connect deployments, and apply to the Analytics, Portal, Management, and Gateway subsystems.

Installing the Automated API behavior testing application

Update the custom resource for installing IBM® API Connect to deploy the Automated API behavior testing application on Kubernetes.

About this task

You can install the Automated API behavior testing application as part of a new IBM API Connect deployment, or you can add it to an existing deployment:

- [Adding the Automated API behavior testing application to an existing deployment](#)
- [Including the Automated API behavior testing application in a new deployment](#)

Note: It is not possible to use the Automated API behavior testing application with a two data center disaster recovery configuration ([Two data center deployment strategy on Kubernetes and OpenShift](#)).

Adding the Automated API behavior testing application to an existing deployment

Before you begin

Make sure that the following requirements are met before attempting to deploy the Automated API behavior testing application:

- The IBM API Connect subsystems are installed.
- The cert-manager is installed for use with IBM API Connect and you are using the default certificates that it generates. The Automated API behavior testing application requires that the IBM API Connect deployment use the cert-manager and the default certificates. Custom certificates are not supported and if used, the Automated API behavior testing application will not install correctly.

About this task

Edit the deployed CR (custom resource) for the Management subsystem and add the settings for the Automated API behavior testing application.

Procedure

1. Retrieve the name of the deployed Management subsystem's deployed CR by running the following command:

```
kubectl get managementcluster -n <management_namespace>
```

2. Edit the deployed CR by running the following command:

```
kubectl edit managementcluster <management-cr-name> -n <management_namespace>
```

3. Append the new `testAndMonitor` definition at the end of the `spec` section, making sure to adhere to the spacing used in the file. Attention: The hub and turnstile endpoints should be distinct and must not overlap with any of the 4 management endpoints.

```
testAndMonitor:
  enabled: true
  hubEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: hub.$STACK_HOST
        secretName: hub-endpoint
  turnstileEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: turnstile.$STACK_HOST
        secretName: turnstile-endpoint
```

`$STACK_HOST` is the ingress subdomain for the API Connect stack, as documented in [Installing the Management subsystem cluster](#).

4. Save the update. The Automated API behavior testing application will be installed into the existing Management subsystem.
5. Check the Hub and Turnstile pods periodically until their "READY" values are both "1/1" and their "STATUS" are both "Running". Verify that the Hub and Turnstile pods are running, by executing the following command:

```
kubectl get pods -n <management_namespace>
```

Look for pods with names similar to `management-hub-XXXX-XXXX` and `management-turnstile-XXXX-XXXX`. The following example shows how the response might look:

NAME	READY	STATUS	RESTARTS	AGE
management-hub-68745dc4b7-gmlkk	1/1	Running	0	29m
management-turnstile-b7d978b56-jghvn	1/1	Running	0	24m

Including the Automated API behavior testing application in a new deployment

Before you begin

Make sure that the cert-manager is installed for use with IBM API Connect and you are using the default certificates that it generates. The Automated API behavior testing application requires that the IBM API Connect deployment use the cert-manager and the default certificates. Custom certificates are not supported and if used, the Automated API behavior testing application will not install correctly.

About this task

Edit the CR (custom resource) for the Management subsystem and add the settings for the Automated API behavior testing application.

Procedure

1. Edit the `ManagementCluster` custom resource (CR) and add the following definition for the Automated API behavior testing application. Append the new definition at the end of the `spec` section, making sure to adhere to the spacing used in the file.

```
testAndMonitor:
  enabled: true
  hubEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: hub.$STACK_HOST
        secretName: hub-endpoint
  turnstileEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
      - name: turnstile.$STACK_HOST
        secretName: turnstile-endpoint
```

`$STACK_HOST` is the ingress subdomain for the API Connect stack, as documented in [Installing the Management subsystem cluster](#).

Attention: The hub and turnstile endpoints should be distinct and must not overlap with any of the 4 management endpoints.

2. Apply the updated CR by running the following command:

```
kubectl apply -f management_cr.yaml -n <management_namespace>
```

where `namespace` is the name of the target installation namespace in the Kubernetes cluster.

The Automated API behavior testing application will be installed with the Management subsystem.

3. Check the Hub and Turnstile pods periodically until their "READY" values are both "1/1" and their "STATUS" are both "Running". Verify that the Hub and Turnstile pods are running, by executing the following command:

```
kubectl get pods -n <management_namespace>
```

Look for pods with names similar to `management-hub-XXXX-XXXX` and `management-turnstile-XXXX-XXXX`. The following example shows how the response might look:

NAME	READY	STATUS	RESTARTS	AGE
management-hub-68745dc4b7-gm1kk	1/1	Running	0	29m
management-turnstile-b7d978b56-jghvn	1/1	Running	0	24m

Installing the toolkit

You can install the toolkit that provides CLI commands, and the API Designer user interface, for IBM® API Connect.

About this task

The toolkit is provided as executable files, so no actual installation is necessary, you just need to download the required compressed file and extract the contents.

There are two toolkit options available:

- CLI: provides a command line environment for working with IBM API Connect.
- CLI + LoopBack + Designer: provides a command line environment for working with IBM API Connect, including LoopBack® support, and the API Designer application.

To install the toolkit, download the compressed file that is appropriate for your chosen toolkit option and platform, then extract the contents to a chosen location on your local machine. The compressed file contains an executable file for running CLI commands and, if you choose the CLI + LoopBack + Designer option, an executable file for launching the API Designer application.

You can download the toolkit from one of the following locations:

- From IBM Fix Central.
- From the Cloud Manager or API Manager user interface.

Procedure

To install and run the toolkit, complete the following steps:

1. Download the toolkit compressed file.
 - To download the toolkit from IBM Fix Central, complete the following steps:
 - Open the [IBM Fix Central site](#) in your browser.
 - In the Product selector field, enter `API Connect`, then select IBM API Connect from the drop down list.
 - Select your Installed Version and click Continue. If you do not know your installed IBM API Connect version, contact your administrator.
 - In the Text field, enter `toolkit`, then click Continue.

- Select the toolkit file that you want to download.
- Click Continue, then follow the instructions to complete the download operation.
- To download the toolkit from Cloud Manager or API Manager user interface, complete the following steps:
 - Open the Cloud Manager or API Manager user interface.
 - On the welcome page, click the Download Toolkit tile. The two toolkit options are listed.
 - Click your platform alongside either the CLI option or the CLI + LoopBack + Designer option as required, then save the associated compressed file to your local file system.
 - To download the API Designer credentials (Client ID and Client Secret), click Download alongside API Designer Credentials, then save the designer_credentials.json file to your local file system. A command is provided for installing the credentials, as described in detail in step 6.
 - To download the toolkit credentials (Client ID and Client Secret), click Download alongside Credential, then save the credentials.json file to your local file system. A command is provided for installing the credentials, as described in detail in step 5.
 - Close the Install API Connect Toolkit window.

2. Extract the contents of the toolkit compressed file to a folder of your choice.

The contents of the file depend on the your chosen toolkit option and platform, as follows:

- The apic-slim or apic-slim.exe file is the CLI for IBM API Connect.
- The apic or apic.exe file is the CLI for IBM API Connect including LoopBack support.

Tip: If you are using the CLI option, then if you rename the apic-slim file to apic, or the apic-slim.exe file to apic.exe, you can run the CLI commands exactly as documented, copy and paste sample commands from the documentation, and use any command scripts as-is if you later move to the CLI + LoopBack + Designer option.
- The api_designer-*platform* file is the API Designer user interface application for the specified platform.

3. Optional: Delete the \$HOME/.apiconnect/node_cli directory.

You need to do this only if you replaced a version of apic or apic.exe and are using the apic 1b4 command. You need to delete the directory get apic to unpack the new loopback.

On Windows, \$HOME is defined by environment USERPROFILE.

4. Run the CLI.

- For the Mac OSX or Linux® platforms, complete the following steps:
 - Open a terminal instance and navigate to the folder where you extracted the contents of the toolkit compressed file.
 - Make the CLI file an executable file by entering the following command:

```
chmod +x download_name
```

Where *download_name* is the name of the toolkit file that you downloaded, either apic or apic-slim.

- Run CLI commands as follows:

```
./apic command_name_and_parameters
```

or

```
./apic-slim command_name_and_parameters
```

For details of the CLI commands, see [apic](#).

- For the Windows platform, complete the following steps:
 - Open a command prompt and navigate to the folder where you extracted the contents of the toolkit compressed file.
 - Run CLI commands as follows:

```
apic command_name_and_parameters
```

or

```
apic-slim command_name_and_parameters
```

For details of the CLI commands, see [apic](#).

Tip: Add the folder location of your CLI file to your PATH variable so that you can run CLI commands from anywhere in your file system.

5. Install the toolkit credentials.

If you do not install the toolkit credentials that are provided for download, as detailed previously in the [toolkit credentials download instructions](#), API Connect uses a default set of credentials that are identical for all deployments. However, the downloaded credentials were generated during the deployment of your API Connect system and are unique to your installation. To install the credentials into your local toolkit, run the following command:

```
apic client-creds:set toolkit_credentials_file_path/credentials.json
```

where *credentials_file_path* is the location to which you downloaded the toolkit credentials JSON file. After you have run this command, your toolkit uses these new credentials to authenticate with the management server.

Note: At any one time, you can use only one set of toolkit credentials for login to a management server from the toolkit CLI. If you want to log in to a different management server you must install the toolkit credentials from that management server.

To revert to the default toolkit credentials for all login operations from the toolkit CLI, use the following command:

```
apic client-creds:clear
```

For increased security, an administrator can remove the default credentials from the management server by completing the following steps:

- Log in to the management server as an administrator; see [Logging in to a management server](#).
- Delete the default credentials by running the following commands:

```
apic registrations:delete toolkit --server mgmt_endpoint_url
apic registrations:delete consumer-toolkit --server mgmt_endpoint_url
```

6. Install the API Designer credentials.

If you do not install the API Designer credentials that are provided for download, as detailed previously in the [API Designer credentials download instructions](#), API Connect uses a default set of credentials that are identical for all deployments. However, the downloaded credentials were generated during the deployment of your API Connect system and are unique to your installation. To install the custom credentials into your local API Designer, set the **APIC_DESIGNER_CREDENTIALS** environment variable to the credentials download location, using the mechanism appropriate for your operating system.

After you have set the **APIC_DESIGNER_CREDENTIALS** environment variable, your API Designer uses the new credentials to authenticate with the management server.

- Windows:

If you are using Windows, create an environment variable using one of the following methods:

- Create a permanent environment variable so that you can start API Designer from any location on your computer:
 - Open the Environment Variables page: Click Start, Settings, System and in the "Related Settings" section of the page, click Advanced System Settings.
 - On the Advanced tab of the System Properties dialog box, click Environment Variables.
 - In the "User variables" section, click New and create an environment variables with the following settings:
 - Variable: **APIC_DESIGNER_CREDENTIALS**
 - Value: `<designer_credentials_file_path>\designer_credentials.json` where `<designer_credentials_file_path>` is the location where you stored the designer_credentials.json file.
 - Click OK to save the new environment variable, and then exit the dialog box.
- If you don't have the required permissions to create a permanent environment variable, you can create a temporary one that will only be used while you are running the API Designer application. The following steps must be performed every time you start API Designer:
 - Open the Windows command prompt.
 - Run the following command to set the temporary environment variable:

```
set APIC_DESIGNER_CREDENTIALS=<designer_credentials_file_path>\designer_credentials.json
```

where `<designer_credentials_file_path>` is the location where you stored the designer_credentials.json file.

Note: Leave the command prompt open for the next step. You must set the temporary variable and start the API Designer within the same Windows command session.

- Now run the following command to start API Designer:

```
C:\Program Files\API Designer\API Designer.exe"
```

By default, API Designer is installed in the C:\Program Files\API Designer folder as shown in the example. If you installed it into a different location, use your own location in the command. Note that the path and file name are enclosed in quotation marks because they contain spaces.

- Mac:

If you are using Mac OS X, create an environment variable using one of the following methods:

- Run the following command to set the global environment variable:

```
launchctl setenv APIC_DESIGNER_CREDENTIALS <designer_credentials_file_path>/designer_credentials.json
```

where `<designer_credentials_file_path>` is the location where you stored the API Designer credentials JSON file.

- Pass in the environment variable while starting API Designer from the command line. With this method, you must run the following command every time you start API Designer:

```
APIC_DESIGNER_CREDENTIALS=<designer_credentials_file_path>/designer_credentials.json open <designer_application_file_path>/'API Designer.app'
```

where:

- `<designer_credentials_file_path>` is the location to which you downloaded the API Designer credentials JSON file.
- `<designer_application_file_path>` is the location to which you downloaded and uncompressed the API Designer application.

7. Start the API Designer application from the location to where you stored the extracted file.

Note:

- To uninstall the API Designer application on a Windows platform with a non Administrator account, complete the following steps:
 - In Windows File Explorer, navigate to the USER_HOME\AppData\Local\Programs\api-designer folder.
 - Run the **Uninstall API Designer application** application. Do **not** use the **Add or remove programs** window.
- To uninstall the API Designer application on a Windows platform with an Administrator account, you can either run the **Uninstall API Designer application** application, or you can use the **Add or remove programs** window.

Results

The IBM API Connect toolkit CLI and, if selected, the API Designer user interface application are installed on your local system.

For information on using the API Designer user interface, see [Developing your APIs and applications](#).

For information on using the toolkit CLI, see [Using the developer toolkit command-line tool](#).

Changing deployment profiles on Kubernetes

You can change an API Connect installation to use different deployment profiles.

About this task

After you complete initial installation of the API Connect subsystems, you can change to another of the profiles that are supported for the subsystem. For more information on the available profiles, see: [Planning your deployment topology](#).

Important considerations:

- If you are changing from a one replica profile to a three replica profile, ensure that you have three replica profile hardware (at least 3 worker nodes).
- Based on the resources configured for each microservice in the profile, there might be downtime during the profile change.

Procedure

1. Management subsystem:

- a. Verify that subsystem is in **Running** state.

```
kubectl get ManagementCluster -n <namespace>
```

- b. Edit the CR:

```
kubectl edit ManagementCluster -n <namespace>
```

Change `spec.profile` value to the new profile you want to use.

- c. When switching from one replica profile to a three replica profile, wait for the microservice to scale up. When switching from a three replica profile to a one replica profile, wait for the microservice to scale down.
d. Verify that subsystem is in **Running** state.

```
kubectl get ManagementCluster -n <namespace>
```

2. Developer Portal subsystem:

- a. Verify that subsystem is in **Running** state.

```
kubectl get PortalCluster -n <namespace>
```

- b. Edit the CR:

```
kubectl edit PortalCluster -n <namespace>
```

Change `spec.profile` value to the new profile you want to use.

- c. When switching from one replica profile to a three replica profile, wait for the microservice to scale up. When switching from a three replica profile to a one replica profile, wait for the microservice to scale down.
d. Verify that subsystem is in **Running** state.

```
kubectl get PortalCluster -n <namespace>
```

3. Analytics subsystem:

- a. Verify that subsystem is in **Running** state.

```
kubectl get AnalyticsCluster -n <namespace>
```

- b. Edit the Analytics CR:

```
kubectl edit AnalyticsCluster -n <namespace>
```

Change `spec.profile` value to the new profile you want to use.

- c. When switching from one replica profile to a three replica profile, wait for the microservice to scale up. When switching from a three replica profile to a one replica profile, wait for the microservice to scale down.
d. Verify that subsystem is in **Running** state.

```
kubectl get AnalyticsCluster -n <namespace>
```

- e. Restart analytics data collection as explained in the topic, [Restarting Analytics pods and data collection](#).

4. Gateway subsystem

- a. Verify that subsystem is in **Running** state.

```
kubectl get GatewayCluster -n <namespace>
```

- b. Edit the CR:

```
kubectl edit GatewayCluster -n <namespace>
```

Change `spec.profile` value to the new profile you want to use.

- c. When switching from one replica profile to a three replica profile, wait for the microservice to scale up. When switching from a three replica profile to a one replica profile, wait for the microservice to scale down.
d. Verify that subsystem is in **Running** state.

```
kubectl get GatewayCluster -n <namespace>
```

Changing deployment profiles on OpenShift

You can change an API Connect installation to use different profiles.

Before you begin

This task assumes that you have already deployed an API Connect cluster as explained in [Deploying on OpenShift and Cloud Pak for Integration](#). If you are changing from a one replica profile to a three replica profile, ensure that you have three replica profile hardware (3 worker nodes) available. For more information on the available profiles, see: [Planning your deployment topology](#).

Procedure

1. Verify that the API Connect cluster is in the **Ready** state by running the following command:

```
oc get apiconnectcluster -n <APIC_namespace>
```

2. Update the API Connect Cluster CR:
 - a. Open the API Connect Cluster CR in edit mode by running the following command:


```
oc edit apiconnectcluster <APIC-instance-name> -n <APIC_namespace>
```
 - b. In the CR, change the `spec.profile` value to the profile that you want to use.

For example, to switch from the one replica profile to the three replica profile, change `n3xc4.m16` to `n12xc4.m12`.
 - c. Save the change by running the following command: `:wq`
3. Wait for microservice to scale up, or scale down, as needed.
4. Verify that the API Connect cluster is in the **Ready** state by running the following command:


```
oc get apiconnectcluster -n <APIC_namespace>
```

Replica override for microservices

Modify the replica count for microservices by using template overrides for Custom Resources.

When API Connect is deployed, the deployment profile you select drives the number of replicas to be used for most deployment. For one replica profiles, specified by `n1x...`, the replica count is 1. For three replica profiles, specified by `n3x...`, replica count is 3. Note, however, that for Cloud Pak for Integration the API Connect Cluster profile `n3xc4.m16` replica count is 1.

You can override the replica count for individual deployments by using template overrides.

- Template overrides are a piece of yml that should be added to the Kubernetes custom resource. The examples provided in this topic show only that the `template:` element should be under the main `spec:` element of a CR. The rest of the CR is not shown.
- Specification of microservice overrides in CP4I requires prefixing the microservice name with a designator for the subsystem type, as follows:
 - Management subsystem: `mgmt-`
 - Gateway: `gw-`
 - Analytics: `a7s-`

For example, for the `apim` Management microservice, use `apim` when defining the override for OpenShift (or Kubernetes), but use `mgmt-apim` for a CP4I override.

Management subsystem

For a Management CR in Kubernetes or OpenShift, the following override specifies 5 replicas for the `apim` deployment:

```
spec:
  template:
    - name: apim
      replicaCount: 5
```

You can also configure replicate count through a similar template override for:

- `juhu`
- `ldap`
- `lur`

The same override for CP4I:

```
spec:
  template:
    - name: mgmt-apim
      replicaCount: 5
```

On CP4I, you can also configure replicate count through a similar template override for:

- `mgmt-juhu`
- `mgmt-ldap`
- `mgmt-lur`

Gateway subsystem

The following override is supported for a Gateway CR in Kubernetes or OpenShift:

```
spec:
  template:
    - name: datapower
      replicaCount: 5
```

The same override for CP4I:

```
spec:
  template:
    - name: gw-datapower
      replicaCount: 5
```

Analytics subsystem

The Analytics microservices that can be scaled using template override are:

- `director`
- `ingestion`
- `mtls-gw`

- **storage**

Note that this microservice cannot be scaled below 2 replicas when using a three replica deployment profile.

On CP4I, you can also configure replica count through a similar template override for:

- **a7s-directory**
- **a7s-ingestion**
- **a7s-mtls-gw**
- **a7s-storage**

Note that this microservice cannot be scaled below 2 replicas when using a three replica deployment profile.

Changing the deployment profile on Cloud Pak for Integration

You can change an API Connect installation to use a different profile.

Before you begin

This task assumes that you have already deployed an API Connect cluster as explained in [Deploying on OpenShift and Cloud Pak for Integration](#). If you are switching from a one replica profile to a three replica profile, ensure that you have three replica profile hardware (3 worker nodes) available. For more information on the available profiles, see: [Planning your deployment topology](#).

Procedure

1. Verify that the API Connect cluster is in the **Ready** state.
Open the IBM Cloud Pak Platform UI, locate the API Management capability on the Integration instances tab, and verify that the state displays as **Ready**. Leave the page open for the next step.
2. Update the API Management instance:
 - a. In the Platform UI, click the overflow menu next to the instance and select Edit.
 - b. In the Details section of the deployment pages, navigate to the Deployment profile field.
 - c. Click the field and select the new deployment profile.
For example, to switch from the one replica profile to the three replica profile, n3xc12.m40.
3. Click Update to save the change.
4. Wait for microservice to scale up, or scale down, as needed.
5. Verify that the status for the API Management instance displays as **Ready**.

Enabling Developer Portal feature flags on Kubernetes

Use a template override to update the installation CR with settings that control the default behavior of the Developer Portal subsystem.

About this task

You can add a template override to the Developer Portal subsystem CR to change the default behavior of the Portal subsystem, or to turn specific features on and off. Table 1 describes the available feature flags.

Table 1. List of available feature flags for the Developer Portal

Feature flag name	Default Value	Pod	Container	Description
<code>PORTAL_SKIP_WEB_ENDPOINT_VALIDATION</code>	<code>true</code>	<code>www</code>	<code>admin</code>	<p>Controls whether the Portal subsystem checks whether the Portal web endpoint is accessible.</p> <p>Sometimes, the endpoint cannot be reached, for example due to the complexity of the networks. In which case the following type of error can be seen in the <code>portal-www</code> pod, and <code>admin</code> container logs:</p> <pre>An error occurred contacting the provided portal web endpoint: example.com The provided Portal web endpoint example.com returned HTTP status code 504</pre> <p>You can disable the check by setting this flag to <code>false</code>.</p>

Feature flag name	Default Value	Pod	Container	Description
ENABLE_TRUSTED_REVERSE_PROXY	false	www	admin	<p>Controls whether the Drupal <code>trusted_reverse_proxy</code> module can manage the reverse proxies list.</p> <p>If you have multiple load balancers and reverse proxies in your upstream network, and the perimeter module is triggered, the incorrect IP address might be banned rather than the intended client IP address. However, you can manage which IP addresses are banned by using the Drupal <code>trusted_reverse_proxy</code> module, and setting this feature flag to <code>true</code>.</p> <p>The Drupal module inspects the <code>x-forwarded-for</code> headers to identify the reverse proxies, and trusts that the first IP address found in the list is the client IP. For example, in <code>X-Forwarded-For: <client>, <proxy1>, <proxy2></code>, the client IP <code><client></code> will be banned. For more information about this module, see https://www.drupal.org/project/trusted_reverse_proxy.</p> <p>To use this module, you must ensure that all of the proxies in your upstream network, including the ingress controller, are configured correctly so that the <code>x-forwarded-for</code> header is passed through to the <code>portal-www</code> pod. For example, the default behavior of the <code>nginx-ingress-controller</code> is to ignore the inbound <code>x-forwarded-for</code> header, and create a new one. To change this behavior, update the <code>nginx-ingress-controller configmap</code> with the following information:</p> <pre>data: compute-full-forwarded-for: "true" use-forwarded-headers: "true"</pre> <p>Warning: Enable this feature flag only if you trust your upstream reverse proxies, as it is possible to trick the <code>x-forwarded-for</code> header.</p>

Procedure

1. If API Connect is already deployed, retrieve the name of the deployed CR for the Developer Portal subsystem by running the following command:

```
kubectl get portalcluster -n <portal_namespace>
```

If API Connect yet is not yet deployed, skip this step.

2. Edit the `portal` CR by running the following command:

```
kubectl edit portalcluster <portal-cr-name> -n <portal_namespace>
```

3. In the editor, append the feature flag definition to the end of the `spec` section, making sure to adhere to the spacing used in the file. The following example enables the `trusted_reverse_proxy` feature.

```
spec:
  ...
  template:
    - containers:
      - env:
        - name: ENABLE_TRUSTED_REVERSE_PROXY
          value: true
          name: container
        name: pod
```

Where:

- `container` is the name of the portal container.
- `pod` is the name of the portal pod.

Note: You can add multiple feature flags to the CR.

4. Save the update. The Developer Portal subsystem is updated to enable or disable features as specified.

Enabling Developer Portal feature flags on OpenShift and Cloud Pak for Integration

Use a template override to update the installation CR with settings that control the default behavior of the Developer Portal subsystem.

About this task

You can add a template override to the Developer Portal subsystem CR to change the default behavior of the Portal subsystem, or to turn specific features on and off. Table 1 describes the available feature flags.

Table 1. List of available feature flags for the Developer Portal

Feature flag name	Default Value	Pod	Container	Description
-------------------	---------------	-----	-----------	-------------

Feature flag name	Default Value	Pod	Container	Description
<code>PORTAL_SKIP_WEB_ENDPOINT_VALIDATION</code>	<code>true</code>	<code>www</code>	<code>admin</code>	<p>Controls whether the Portal subsystem checks whether the Portal web endpoint is accessible.</p> <p>Sometimes, the endpoint cannot be reached, for example due to the complexity of the networks. In which case the following type of error can be seen in the <code>portal-www</code> pod, and <code>admin</code> container logs:</p> <pre>An error occurred contacting the provided portal web endpoint: example.com The provided Portal web endpoint example.com returned HTTP status code 504</pre> <p>You can disable the check by setting this flag to <code>false</code>.</p>
<code>ENABLE_TRUSTED_REVERSE_PROXY</code>	<code>false</code>	<code>www</code>	<code>admin</code>	<p>Controls whether the Drupal <code>trusted_reverse_proxy</code> module can manage the reverse proxies list.</p> <p>If you have multiple load balancers and reverse proxies in your upstream network, and the perimeter module is triggered, the incorrect IP address might be banned rather than the intended client IP address. However, you can manage which IP addresses are banned by using the Drupal <code>trusted_reverse_proxy</code> module, and setting this feature flag to <code>true</code>.</p> <p>The Drupal module inspects the <code>x-forwarded-for</code> headers to identify the reverse proxies, and trusts that the first IP address found in the list is the client IP. For example, in <code>x-Forwarded-For: <client>, <proxy1>, <proxy2></code>, the client IP <code><client></code> will be banned. For more information about this module, see https://www.drupal.org/project/trusted_reverse_proxy.</p> <p>To use this module, you must ensure that all of the proxies in your upstream network, including the ingress controller, are configured correctly so that the <code>x-forwarded-for</code> header is passed through to the <code>portal-www</code> pod. For example, the default behavior of the <code>nginx-ingress-controller</code> is to ignore the inbound <code>x-forwarded-for</code> header, and create a new one. To change this behavior, update the <code>nginx-ingress-controller configmap</code> with the following information:</p> <pre>data: compute-full-forwarded-for: "true" use-forwarded-headers: "true"</pre> <p>Warning: Enable this feature flag only if you trust your upstream reverse proxies, as it is possible to trick the <code>x-forwarded-for</code> header.</p>

Procedure

1. If API Connect is already deployed, retrieve the name of the deployed CR by running the following command:

```
oc get apiconnectcluster -n <APIC_namespace>
```

If API Connect yet is not yet deployed, skip this step.

2. Edit the `apiconnectcluster` CR by running the following command:

```
oc edit apiconnectcluster <APIC_instance_name> -n <APIC_namespace>
```

3. In the CR, append the feature flag definition to the end of the `spec.portal` section, making sure to adhere to the spacing used in the file. The following example enables the `trusted_reverse_proxy` feature.

```
spec:
  ...
  portal:
    template:
      - containers:
        - env:
          - name: ENABLE_TRUSTED_REVERSE_PROXY
            value: true
            name: container
          name: pod
```

Where:

- `container` is the name of the portal container.
- `pod` is the name of the portal pod.

Note: You can add multiple feature flags to the CR.

4. Close and save the file with the following command:

```
:wq
```

The Developer Portal subsystem is updated to enable or disable features as specified.

Enabling UI feature flags on Kubernetes

Use a template override to update the installation CR with settings that control UI features.

About this task

You can add a template override to the Management subsystem CR and use it to control whether specific features are available in the UI. Some features can be optionally disabled by users (by disabling them in the browser) provided you have enabled the feature for the cluster as a whole. Table 1 describes the available feature flags.

Table 1. Flags and settings for controlling UI features

Feature ID	Default Value	Partial Override For	Description
<code>ffAdvancedPublish</code>	<code>true</code>		Enable advanced publish option (deprecated)
<code>allOmniSearch</code>	<code>true</code>		Enables OmniSearch for all lists. This flag is used in conjunction with other flags.
<code>draftsOmniSearch</code>	<code>true</code>	<code>allOmniSearch</code>	Enable OmniSearch for drafts list
<code>catalogProductsOmniSearch</code>	<code>true</code>	<code>allOmniSearch</code>	Enable OmniSearch for catalog
<code>orgsOmniSearch</code>	<code>true</code>	<code>allOmniSearch</code>	Enable OmniSearch for org lists
<code>appsOmniSearch</code>	<code>true</code>	<code>allOmniSearch</code>	Enable OmniSearch for apps
<code>subscriptionsOmniSearch</code>	<code>true</code>	<code>allOmniSearch</code>	Enable OmniSearch for subscriptions, if disabled, subscription lists will be removed from catalog list
<code>requestedApprovalsOmniSearch</code>	<code>true</code>	<code>allOmniSearch</code>	Enable OmniSearch for requested approvals
<code>approvalTasksOmniSearch</code>	<code>true</code>	<code>allOmniSearch</code>	Enable OmniSearch for approvals
<code>productApisOmniSearch</code>	<code>true</code>	<code>allOmniSearch</code>	Enable OmniSearch for apis within products
<code>membersOmniSearch</code>	<code>true</code>	<code>allOmniSearch</code>	Enable OmniSearch for member lists

Procedure

1. If you already deployed API Connect, retrieve the name of the deployed CR for the Management subsystem by running the following command:

```
kubectl get managementcluster -n <management_namespace>
```

If you did not deploy API Connect yet, skip this step.

2. Edit the `management` CR by running the following command:

```
kubectl edit managementcluster <management-cr-name> -n <management_namespace>
```

3. In the editor, append the feature-flag definition to the end of the `spec:` section, making sure to adhere to the spacing used in the file. The following example enables the OmniSearch feature on all possible UI pages.

```
spec:
  template:
    - name: ui
      containers:
        - name: ui
          env:
            - name: APIC_UI_FEATURES
              value: allOmniSearch=always
```

`APIC_UI_FEATURES` is a URL-encoded string. Each parameter is formed as `featureId=$featureValue`, and you can list multiple parameters with the `&` delimiter (`featureId=$featureValue&featureId=$featureValue`).

Use the `featureId` values from Table 1. For each feature, specify one of the following `featureValue` settings:

- `true` - On by default, but can be overridden by the user, for their browser alone
- `false` - Off by default, but can be overridden by the user, for their browser alone
- `never` - Off regardless of browser setting, disabled in feature flag page
- `always` - On regardless of browser setting, disabled in feature flag page

4. Save the update.

The cluster is updated to enable or disable features as specified.

Enabling UI feature flags on OpenShift and Cloud Pak for Integration

Use a template override to update the installation CR with settings that control UI features.

About this task

You can add a template override to the Management subsystem CR and use it to control whether specific features are available in the UI. Some features can be optionally disabled by users (by disabling them in the browser) provided you have enabled the feature for the cluster as a whole. Table 1 describes the available feature flags.

Table 1. Flags and settings for controlling UI features

Feature ID	Default Value	Partial Override For	Description
<code>ffAdvancedPublish</code>	<code>true</code>		Enable advanced publish option (deprecated)
<code>allOmniSearch</code>	<code>true</code>		Enables OmniSearch for all lists. This flag is used in conjunction with other flags.
<code>draftsOmniSearch</code>	<code>true</code>	<code>allOmniSearch</code>	Enable OmniSearch for drafts list
<code>catalogProductsOmniSearch</code>	<code>true</code>	<code>allOmniSearch</code>	Enable OmniSearch for catalog
<code>orgsOmniSearch</code>	<code>true</code>	<code>allOmniSearch</code>	Enable OmniSearch for org lists
<code>appsOmniSearch</code>	<code>true</code>	<code>allOmniSearch</code>	Enable OmniSearch for apps

Feature ID	Default Value	Partial Override For	Description
subscriptionsOmniSearch	true	allOmniSearch	Enable OmniSearch for subscriptions, if disabled, subscription lists will be removed from catalog list
requestedApprovalsOmniSearch	true	allOmniSearch	Enable OmniSearch for requested approvals
approvalTasksOmniSearch	true	allOmniSearch	Enable OmniSearch for approvals
productApisOmniSearch	true	allOmniSearch	Enable OmniSearch for apis within products
membersOmniSearch	true	allOmniSearch	Enable OmniSearch for member lists

Procedure

1. If you already deployed API Connect, retrieve the name of the deployed CR by running the following command:

```
oc get apiconnectcluster -n <APIC_namespace>
```

If you did not deploy API Connect yet, skip this step.

2. Edit the `apiconnectcluster` CR by running the following command:

```
oc edit apiconnectcluster <APIC_instance_name> -n <APIC_namespace>
```

3. In the CR, append the feature-flag definition to the end of the `spec.management` section, making sure to adhere to the spacing used in the file. The following example enables the OmniSearch feature on all possible UI pages.

```
spec:
  management:
    template:
      - name: ui
        containers:
          - name: ui
            env:
              - name: APIC_UI_FEATURES
                value: allOmniSearch=always
```

`APIC_UI_FEATURES` is a URL-encoded string. Each parameter is formed as `featureId=$featureValue`, and you can list multiple parameters with the `&` delimiter (`featureId=$featureValue&featureId=$featureValue`).

Use the `featureId` values from Table 1. For each feature, specify one of the following `featureValue` settings:

- **true** - On by default, but can be overridden by the user, for their browser alone
- **false** - Off by default, but can be overridden by the user, for their browser alone
- **never** - Off regardless of browser setting, disabled in feature flag page
- **always** - On regardless of browser setting, disabled in feature flag page

4. Close and save the file with the following command:

```
:wq
```

The cluster is updated to enable or disable features as specified.

Advanced configuration for the management subsystem

Specify advanced configuration for the management subsystem.

See:

- [Configuring maximum size of client requests to the Management subsystem](#)
You can configure the maximum allowed size of the client request body for requests made to the Management subsystem.
- [Setting rate limits for public APIs on the management service for a Kubernetes environment](#)
Describes the procedure for setting a rate limit for public APIs on the management service. Rate limits provide protection from DDoS (distributed denial of service) attacks.
- [Configuring monetization on Kubernetes](#)
You can optionally configure monetization in API Connect on a Kubernetes deployment by enabling the billing microservice.
- [Overriding resources for Postgres components on Kubernetes](#)
You can optionally override the CPU and Memory resources used by postgres database components.
- [Overriding resources for Postgres components on OpenShift](#)
You can optionally override the CPU and Memory resources used by postgres database components.

Configuring maximum size of client requests to the Management subsystem

You can configure the maximum allowed size of the client request body for requests made to the Management subsystem.

About this task

The default maximum size is 8 megabytes. You can increase this value.

Increasing the setting can be useful if you get errors when using the CLI to import a large API. Example error:

Procedure

1. Edit the Management CR. Refer to the Management CR templates:

```
spec:
  ...
  ...
  template:
  - name: juhu
    containers:
    - name: juhu
      env:
      - name: CLIENT_MAX_BODY_SIZE
        value: "12m"
```

Note that the default value is "8m". The `template:` entry must be indented two spaces, and be located within the `spec:` element.

2. Apply the changes:

```
kubectl -n <namespace> apply -f management_cr.yaml
```

Setting rate limits for public APIs on the management service for a Kubernetes environment

Describes the procedure for setting a rate limit for public APIs on the management service. Rate limits provide protection from DDoS (distributed denial of service) attacks.

Before you begin

Note: This article refers to third-party software that IBM does not control. As such, the software may change and this information may become outdated. These instructions assume you have a working Kubernetes environment with the `kubectl` command-line tool installed, and that you understand how to manage Kubernetes. For more information, see <https://kubernetes.io>.

About this task

Rate limits can be set for public APIs on the management service. Rate limits on APIs help provide protection from DDoS (distributed denial of service) attacks. Without a rate limit, API calls from public APIs are unlimited.

The rate limit configuration requires that the header contains the actual client IP address. Any load balancer or proxy (for example, HAProxy) that is installed in front of the management service must be configured to pass the actual client IP address.

This procedure must be performed on a running API Connect deployment.

Rate limits are calculated as requests per seconds per client.

Note: If the rate limit has been reached on the management subsystem, the client will get an HTTP error: **429 Too Many Requests**.

Procedure

- Configure the `ingress-nginx-ingress-controller` to pass the real client IP address:
 1. Set the `use-proxy-protocol` parameter to `true` by entering the following command:

```
kubectl edit ConfigMap -n kube-system ingress-nginx-ingress-controller
```

2. In the ConfigMap, set the following: `use-proxy-protocol:`
`"true"`

- Set a rate limit:
 1. You can add a `template:` value in your subsystem CR to provide an override. For example, the Management CR template value can be:

```
spec:
  ...
  template:
  - name: juhu
    containers:
    - name: juhu
      env:
      - name: RATE_LIMIT_PER_CLIENT
        value: "10"
      - name: LIMIT_REQUEST_OPTION
        value: "burst=10 nodelay"
```

Note that the `template:` entry must be indented two spaces, and must be located in the `spec:` element.

In this example, the first option sets the rate to 10 requests per second (10r/s). The second option allows 10 requests boosted without delay within < 1 second. You can customize as needed.

The `rateLimitPerClient` property sets `rate`, and `limitRequestOption` sets `[burst=number] [nodelay | delay=number]` in the following nginx configuration:

```
limit_req zone key zone=name:size rate=rate;  
limit_req zone=name [burst=number] [nodelay | delay=number];
```

Note that `zone` has been pre-defined and can't be configured. For more details, see the nginx doc http://nginx.org/en/docs/http/ngx_http_limit_req_module.html.

2. Apply the changes:

```
kubectl -n <namespace> apply -f management_cr.yaml
```

3. Validate that the juhu pod has restarted by listing the pods:

```
kubectl get pods -n <namespace> | grep juhu
```

4. Check the AGE column to ensure a new juhu pod has started.

5. If the older juhu pod is still running, delete it with the following command:

```
kubectl delete pods -n <namespace> <old-juhu-pod>
```

• Disable a rate limit:

1. Validate that the juhu pod has restarted by listing the pods:

```
kubectl get pods -n <namespace> | grep juhu
```

2. Check the AGE column to ensure a new juhu pod has started.

3. If the older juhu pod is still running, delete it with the following command:

```
kubectl delete pods -n <namespace> <old-juhu-pod>
```

Configuring monetization on Kubernetes

You can optionally configure monetization in API Connect on a Kubernetes deployment by enabling the billing microservice.

About this task

API Connect includes a subscription billing microservice that allows API providers to define pricing plans in their API Products, and monetize their API offerings. If a Product contains a pricing plan, API Consumers must enter their payment information into the Developer Portal before they can subscribe to that plan. API Connect supports integration with Stripe Subscription Billing, an independent cloud service that manages monetized product plans, customers, their payment information, and their subscription history, in order to generate monthly invoices and charge customers automatically. With this integration, Stripe serves as both the subscription billing system and the payment processing system.

Note:

- These instructions apply only to Kubernetes installations. For VMware installations, see [Configuring monetization on VMware](#).
- To use Stripe as your credit card processing vendor, you must have port 443 open to HTTPS communication between the Stripe API and your Developer Portal management and the Management cluster servers. See [Firewall requirements on Kubernetes](#) and [Firewall requirements on VMware](#) for more information about this requirement.
- Provider organizations cannot add any billing integrations until the billing microservice is enabled.
- After provider organizations have added billing integrations, the billing microservice must not be disabled.

To enable or disable the billing microservice, you must configure the Management subsystem custom resource (CR) file. See the following instructions:

- [Enabling the billing microservice as part of a new deployment](#)
- [Enabling the billing microservice as part of an existing deployment](#)
- [Disabling the billing microservice](#)

Procedure

- **Enabling the billing microservice as part of a new deployment**

Edit the CR file for the Management subsystem and add the settings for the billing microservice.

1. Edit the `ManagementCluster` CR and add the following definition for the billing microservice. Append the billing definition to the end of the `spec:` section, making sure to adhere to the spacing used in the file.

```
spec:  
  ...  
  billing:  
    enabled: true
```

2. Apply the updated CR by running the following command as part of the standard Management subsystem installation (see [Installing the Management subsystem cluster](#) for details):

```
kubectl apply -f management_cr.yaml -n <management_namespace>
```

Where `management_namespace` is the name of the target installation namespace in the Kubernetes cluster. The billing microservice will be enabled with the Management subsystem.

3. You can monitor your Kubernetes deployments by running the following command:

```
kubectl get deployments -n <management_namespace>
```

The installation is complete when the `management-billing` pod is shown in the list of returned values.

After the billing microservice is enabled, provider organization owners can create billing integration resources for API providers in the API Manager UI. For more information, see [Monetizing your Products](#).

- **Enabling the billing microservice as part of an existing deployment**

Edit the deployed CR for the Management subsystem and add the settings for the billing microservice.

1. Retrieve the name of the deployed CR for the Management subsystem by running the following command:

```
kubectl get managementcluster -n <management_namespace>
```

Where *management_namespace* is the name of the target installation namespace in the Kubernetes cluster.

2. Edit the deployed CR by running the following command:

```
kubectl edit managementcluster <management-cr-name> -n <management_namespace>
```

Where:

- *management-cr-name* is the name of the deployed CR for the Management subsystem.
- *management_namespace* is the name of the target installation namespace in the Kubernetes cluster.

3. In the editor, append the billing definition to the end of the **spec:** section, making sure to adhere to the spacing used in the file.

```
spec:
  ...
  billing:
    enabled: true
```

4. Save the update.

The billing microservice is enabled in the Management subsystem.

5. You can monitor your Kubernetes deployments by running the following command:

```
kubectl get deployments -n <management_namespace>
```

The installation is complete when the **management-billing** pod is shown in the list of returned values.

After the billing microservice is enabled, provider organization owners can create billing integration resources for API providers in the API Manager UI. For more information, see [Monetizing your Products](#).

- **Disabling the billing microservice**

Important: After provider organizations have added billing integrations, the billing microservice must not be disabled.

Edit the deployed CR for the Management subsystem and update the settings for the billing microservice.

1. Retrieve the name of the deployed CR for the Management subsystem by running the following command:

```
kubectl get managementcluster -n <management_namespace>
```

Where *management_namespace* is the name of the target installation namespace in the Kubernetes cluster.

2. Edit the deployed CR by running the following command:

```
kubectl edit managementcluster <management-cr-name> -n <management_namespace>
```

Where:

- *management-cr-name* is the name of the deployed CR for the Management subsystem.
- *management_namespace* is the name of the target installation namespace in the Kubernetes cluster.

3. In the editor, change the billing definition to **enabled:**

```
false.

spec:
  ...
  billing:
    enabled: false
```

4. Save the update.

The billing microservice is disabled in the Management subsystem.

Related information

- [Monetizing your Products](#)
- [stripe.com](#)

Overriding resources for Postgres components on Kubernetes

You can optionally override the CPU and Memory resources used by postgres database components.

About this task

There are three postgres components:

- **postgres**
- **pgBouncer**
- **pgBackRest**

The CPU, memory, and PVC resources used by the **postgres**, **pgBouncer**, and **pgBackRest** components depend on the deployment profile that was selected on installation. The resources that are defined for each deployment profile are shown here: [Management component deployment profile limits](#).

Important: The postgres pod is licensed based on CPU use, so adjusting this pod's CPU limit has licensing implications, see: [Component profiles, CPU limits, memory limits, and licensing](#)

Procedure

To override resources, complete the following procedure

1. Prepare a patch yaml.

- Ensure that you specify the name of the container for the resource you want to override. Names:
 - `postgres`
 - `pgBouncer`
 - `pgBackRest`
- You can override multiple postgres components in parallel. Adding one container section under `pgcluster` template.
- You can override just a single value, such as CPU limit.

Examples:

- Example yaml, to replace all postgres DB resources:

```
spec:
  template:
    - name: pgcluster
      containers:
        - name: postgres
          resources:
            requests:
              cpu: 1500m
              memory: 8Gi
            limits:
              cpu: 600m
              memory: 512Mi
```

- Example yaml, to replace all pgBouncer resources:

```
spec:
  template:
    - name: pgcluster
      containers:
        - name: pgBouncer
          resources:
            requests:
              cpu: 200m
              memory: 128Mi
            limits:
              cpu: 300m
              memory: 256Mi
```

- Example yaml, to replace CPU limit of `postgres` database:

```
spec:
  template:
    - name: pgcluster
      containers:
        - name: postgres
          resources:
            limits:
              cpu: 1000m
```

- Example yaml, to replace CPU limits of `postgres` DB and `pgBouncer`:

```
spec:
  template:
    - name: pgcluster
      containers:
        - name: postgres
          resources:
            limits:
              cpu: 1000m
        - name: pgBouncer
          resources:
            limits:
              cpu: 1000m
```

2. Apply the patch yaml:

```
kubectl patch mgmt <mgmt-cr-name> --type merge --patch "$(cat patch.yaml)"
```

Results

The postgres components will restart with updated values.

Overriding resources for Postgres components on OpenShift

You can optionally override the CPU and Memory resources used by postgres database components.

About this task

There are three postgres components:

- `postgres`
- `pgBouncer`
- `pgBackRest`

The CPU, memory, and PVC resources used by the `postgres`, `pgBouncer`, and `pgBackRest` components depend on the deployment profile that was selected on installation. The resources that are defined for each deployment profile are shown here: [Management component deployment profile limits](#). Important: The postgres pod is licensed based on CPU use, so adjusting this pod's CPU limit has licensing implications, see: [Component profiles, CPU limits, memory limits, and licensing](#)

Procedure

To override resources, complete the following procedure

1. Prepare a patch yaml.

- Ensure that you specify the name of the container for the resource you want to override. Names:

- `postgres`
- `pgBouncer`
- `pgBackRest`

- You can override multiple postgres components in parallel. Adding one container section under `mgmt-pgcluster` template.
- You can override just a single value, such as CPU limit.

Examples:

- Example yaml, to replace all postgres DB resources:

```
spec:
  template:
  - name: mgmt-pgcluster
    containers:
    - name: postgres
      resources:
        requests:
          cpu: 1500m
          memory: 8Gi
        limits:
          cpu: 600m
          memory: 512Mi
```

- Example yaml, to replace all `pgBouncer` resources:

```
spec:
  template:
  - name: mgmt-pgcluster
    containers:
    - name: pgBouncer
      resources:
        requests:
          cpu: 200m
          memory: 128Mi
        limits:
          cpu: 300m
          memory: 256Mi
```

- Example yaml, to replace CPU limit of `postgres` database:

```
spec:
  template:
  - name: mgmt-pgcluster
    containers:
    - name: postgres
      resources:
        limits:
          cpu: 1000m
```

- Example yaml, to replace CPU limits of `postgres` DB and `pgBouncer`:

```
spec:
  template:
  - name: mgmt-pgcluster
    containers:
    - name: postgres
      resources:
        limits:
          cpu: 1000m
    - name: pgBouncer
      resources:
        limits:
          cpu: 1000m
```

2. Apply the patch yaml:

```
oc patch apiconnectcluster <apiconnectcluster-cr-name> --type merge --patch "$(cat patch.yaml)"
```

Results

The postgres components will restart with updated values.

Advanced configuration for the analytics subsystem

Specify advanced configuration for the analytics subsystem on Kubernetes, OpenShift, or Cloud Pak for Integration.

- [Changing storage type between shared and dedicated](#)
How to change from dedicated analytics storage to shared, or from shared to dedicated.
- [Horizontal scaling of analytics data storage pod replicas](#)
When analytics dedicated storage is used, you can scale your data storage pod replicas for increased API event data storage and throughput.

Changing storage type between shared and dedicated

How to change from dedicated analytics storage to shared, or from shared to dedicated.

If you have a three replica deployment and want to change storage type, follow the steps in this topic.

Note: Dedicated storage is not available on a one replica deployment.

For more information about dedicated storage, see [Planning the analytics profile, storage class, and storage type](#).

Note: For OpenShift users: The example steps in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place. If you are using a top-level CR you must edit the API Connect cluster CR (the top-level CR), instead of directly in the subsystem CRs. If the subsystem section is not included in the top-level CR, copy and paste the section from the subsystem CR to the top-level CR.

Changing from shared to dedicated storage

Edit your analytics CR, and change storage type to dedicated:

```
kubectl edit analyticscluster -n <namespace>
```

```
...
storage:
  type: dedicated
```

If your environment does not have a default storage class, or want to use a different storage class for your `storage-os-master` pods, then also add the `spec.storage.master` section to specify the storage class, for example:

```
storage:
  type: dedicated
...
master:
  volumeClaimTemplate:
    storageClassName: local_storage
```

Changing from dedicated to shared storage

1. Edit your analytics CR, and change storage type to shared:

```
kubectl edit analyticscluster -n <namespace>
```

```
...
storage:
  type: shared
```

2. Wait until all of your `storage` pods restart and are in the `Ready` state.
3. Edit your analytics CR and remove the `spec.storage.master` section.
4. Delete the `data-analytics-storage-os-master` PVCs:

```
kubectl delete pvc data-analytics-storage-os-master-<integer> -n <namespace>
```

Horizontal scaling of analytics data storage pod replicas

When analytics dedicated storage is used, you can scale your data storage pod replicas for increased API event data storage and throughput.

To take advantage of analytics storage pod scaling, you must have a three replica deployment, and be using dedicated storage, see [Planning the analytics profile, storage class, and storage type](#).

Note:

- The term "replica" when used with data storage pod refers to the Kubernetes storage pod, not OpenSearch replicas, which are not configurable in API Connect.
- The OpenSearch term "node" refers to OpenSearch data storage nodes. OpenSearch nodes have a one-to-one relationship with the API Connect storage pod replicas, and use the same name.

This operation requires the use of the Analytics CLI. [Using the Analytics CLI](#).

Increase number of data storage pod replicas

1. Edit your analytics CR, and add the following to the `spec` section:

```
template:
- name: storage
  replicaCount: <new replica count>
```


where `<new replica count>` must be less than or equal to the number of worker nodes you have available.

2. Wait for the new storage pod replicas to start on your new worker nodes, and for the analytics cluster to rebalance your API event data to use your new replicas. The time the rebalancing takes depends on the size of your API event data and the network transmission speed between worker nodes. You can check the rebalancing status with the CLI health command:

```
apic --mode analytics clustermgmt:getHealth --server <management server platform api endpoint> --analytics-service <analytics service name> --format json
```

where:

- `<management server platform api endpoint>` is the management subsystem platform API endpoint.
- `<analytics service name>` is the name of the analytics service as configured in the Cloud Manager UI.

When rebalancing is complete, the `number_of_data_nodes` matches your updated `replicaCount`, and the highlighted `_shards` properties are all zero, as shown:

```
{
  "cluster_name": "apic-analytics-cluster",
  "status": "green",
  "timed_out": false,
  "number_of_nodes": 8,
  "number_of_data_nodes": 5,
  "discovered_master": true,
  "discovered_cluster_manager": true,
  "active_primary_shards": 8,
  "active_shards": 24,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 0,
  "delayed_unassigned_shards": 0,
  "number_of_pending_tasks": 0,
  "number_of_in_flight_fetch": 0,
  "task_max_waiting_in_queue_millis": 0,
  "active_shards_percent_as_number": 100
}
```

Note: If in the past you increased and then reduced the number of data storage replicas in your deployment, you might have an OpenSearch exclude list that contains the names of the previous replicas you removed. New storage pod replicas cannot be created if their name is in the OpenSearch exclude list. To empty the OpenSearch exclude list, use the CLI to set the list to an empty string:

1. Create a JSON file called `empty_exclude_list.json`, and paste in the following contents:

```
{ "persistent.cluster.routing.allocation.exclude._name" : "" }
```

2. Apply the exclude list to your OpenSearch cluster. Use the `clustermgmt:putSettings` CLI command with the JSON file created in step 1.

```
apic -m analytics clustermgmt:putSettings --analytics-service <analytics service name> --server <management server platform api endpoint> empty_exclude_list.json --format json
```

The command returns showing that the list is set to an empty string:

```
...
  "persistent": {
    "cluster": {
      "routing": {
        "allocation": {
          "exclude": {
            "_name": ""
          }
        }
      }
    }
  }
...

```

Reduce number of data storage pod replicas

1. Verify that your analytics cluster health is green. Use the analytics CLI health command:

```
apic --mode analytics clustermgmt:getHealth --server <management server platform api endpoint> --analytics-service <analytics service name> --format json
```

where:

- `<management server platform api endpoint>` is the management subsystem platform API endpoint.
- `<analytics service name>` is the name of the analytics service as configured in the Cloud Manager UI.

Check that the status property is green:

```
{
  "cluster_name": "apic-analytics-cluster",
  "status": "green",
  "timed_out": false,
  "number_of_nodes": 8,
  "number_of_data_nodes": 5,
  "discovered_master": true,
  "discovered_cluster_manager": true,
  "active_primary_shards": 8,
  "active_shards": 24,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 0,
  "delayed_unassigned_shards": 0,
  "number_of_pending_tasks": 0,
  "number_of_in_flight_fetch": 0,
  "task_max_waiting_in_queue_millis": 0,
  "active_shards_percent_as_number": 100
}
```

Do not proceed if the status is not green.

2. Replicas are removed by reducing the Kubernetes `replicaCount` in the analytics CR. When `replicaCount` is reduced, the replicas that are deleted are always the highest numbered replicas. For example:

```
kubectl get pods | grep storage
```

NAME	READY	STATUS	RESTARTS	AGE
analytics-storage-0	1/1	Running	0	140d
analytics-storage-1	1/1	Running	0	140d
analytics-storage-2	1/1	Running	0	140d
analytics-storage-3	1/1	Running	0	140d
analytics-storage-4	1/1	Running	0	140d

If you reduce `replicaCount` from 5 to 3, the two replicas that are deleted are `analytics-storage-3` and `analytics-storage-4`.

Before you reduce the `replicaCount`, you must ensure that replicas to be deleted contain no API event data. To make OpenSearch relocate all API event data to the replicas you are keeping, add the replicas that you want to delete to the OpenSearch exclude list:

- a. Create a JSON file called `replicas_exclude_list.json`, and paste in the following contents:

```
{ "persistent.cluster.routing.allocation.exclude._name" : "<replicas to remove>" }
```

where `<replicas to remove>` is a comma-separated list of the replicas to be removed, for example:

```
{ "persistent.cluster.routing.allocation.exclude._name" : "analytics-storage-3,analytics-storage-4" }
```

- b. Apply the exclude list to your OpenSearch cluster. Use the `clustermgmt:putSettings` CLI command with the JSON file created in step [2.a](#):

```
apic -m analytics clustermgmt:putSettings --analytics-service <analytics service name> --server <management server platform api endpoint> replica_exclude_list.json --format json
```

The command returns confirmation that the list is updated with what is in `replicas_exclude_list.json`:

```
...
  "persistent": {
    "cluster": {
      "routing": {
        "allocation": {
          "exclude": {
            "_name": "<replicas to remove>"
          }
        }
      }
    }
  }
...

```

3. Wait for your analytics cluster to remove all data from the replicas you want to delete, which you specified in step [2](#). You can check that the data rebalancing is complete with the `clustermgmt:catAllocation` CLI command:

```
apic -m analytics clustermgmt:catAllocation --analytics-service <analytics service name> --server <management server platform api endpoint> --format json --return_format json
```

Replicas that don't have data appear as entries with a `shards` value of zero in the output (you can ignore the `disk.*` properties):

```
[
  {
    "shards": "5",
    "disk.indices": "113.3kb",
    "disk.used": "27.3gb",
    "disk.avail": "222.5gb",
    "disk.total": "249.8gb",
    "disk.percent": "10",
    "host": "192.168.87.226",
    "ip": "192.168.87.226",
    "node": "analytics-storage-0"
  },
  {
    "shards": "0",
    "disk.indices": "0",
    "disk.used": "36.9kb",
    "disk.avail": "212.9kb",
    "disk.total": "249.8kb",
    "disk.percent": "1",
    "host": "192.168.64.225",
    "ip": "192.168.64.225",
    "node": "analytics-storage-3"
  },
  {
    "shards": "5",
    "disk.indices": "16.8kb",
    "disk.used": "28.2gb",
    "disk.avail": "221.6gb",
    "disk.total": "249.8gb",
    "disk.percent": "11",
    "host": "192.168.112.227",
    "ip": "192.168.112.227",
    "node": "analytics-storage-1"
  },
  {
    "shards": "5",
    "disk.indices": "23.3kb",
    "disk.used": "29.3gb",
    "disk.avail": "220.4gb",
    "disk.total": "249.8gb",
    "disk.percent": "11",
    "host": "192.168.24.102",
    "ip": "192.168.24.102",
    "node": "analytics-storage-2"
  },
]
```

```
{
  "shards": "0",
  "disk.indices": "0",
  "disk.used": "35.4kb",
  "disk.avail": "214.4kb",
  "disk.total": "249.8kb",
  "disk.percent": "1",
  "host": "192.168.91.149",
  "ip": "192.168.91.149",
  "node": "analytics-storage-4"
}
]
```

The time rebalancing takes depends on the size of your API event data and the network transmission speed between pod replicas.

Note: The value in the `node` property of the `clustermgmt:catAllocation` output is the same as the names of the Kubernetes data storage pod replicas.

4. Verify that your analytics cluster health is green. Use the analytics CLI health command:

```
apic --mode analytics clustermgmt:getHealth --server <management server platform api endpoint> --analytics-service <analytics service name> --format json
```

where:

- `<management server platform api endpoint>` is the management subsystem platform API endpoint.
- `<analytics service name>` is the name of the analytics service as configured in the Cloud Manager UI.

Check that the status property is green:

```
{
  "cluster_name": "apic-analytics-cluster",
  "status": "green",
  "timed_out": false,
  "number_of_nodes": 8,
  "number_of_data_nodes": 5,
  "discovered_master": true,
  "discovered_cluster_manager": true,
  "active_primary_shards": 8,
  "active_shards": 24,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 0,
  "delayed_unassigned_shards": 0,
  "number_of_pending_tasks": 0,
  "number_of_in_flight_fetch": 0,
  "task_max_waiting_in_queue_millis": 0,
  "active_shards_percent_as_number": 100
}
```

Do not proceed if the status is not green.

5. Edit your analytics CR, and update the `replicaCount`:

```
template:
- name: storage
  replicaCount: <new replica count>
```

where `<new replica count>` is the reduced number of replicas that you want.

Note: If the `replicaCount` property does not exist in the analytics CR, check that you are not attempting to reduce `replicaCount` to 2. Reducing `replicaCount` to less than 3 is not a supported operation.

6. Monitor the health status of your analytics cluster with the CLI health command:

```
apic --mode analytics clustermgmt:getHealth --server <management server platform api endpoint> --analytics-service <analytics service name> --format json
```

where:

- `<management server platform api endpoint>` is the management subsystem platform API endpoint.
- `<analytics service name>` is the name of the analytics service as configured in the Cloud Manager UI.

Confirm that the status is green, the `number_of_data_nodes` matches your updated `replicaCount`, and that the highlighted shard properties are all zero:

```
{
  "cluster_name": "apic-analytics-cluster",
  "status": "green",
  "timed_out": false,
  "number_of_nodes": 6,
  "number_of_data_nodes": 3,
  "discovered_master": true,
  "discovered_cluster_manager": true,
  "active_primary_shards": 8,
  "active_shards": 24,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 0,
  "delayed_unassigned_shards": 0,
  "number_of_pending_tasks": 0,
  "number_of_in_flight_fetch": 0,
  "task_max_waiting_in_queue_millis": 0,
  "active_shards_percent_as_number": 100
}
```

Advanced configuration for the gateway subsystem

Specify advanced configuration for the gateway subsystem.

See:

- [Customizing a DataPower deployment](#)
Update the `GatewayCluster` CR to configure `DataPowerService` APIs to customize your DataPower deployment.
- [Overriding the default DataPower image, version, and license](#)
Update the `GatewayCluster` CR to override the image, version, and license information for the Gateway subsystem.
- [Enabling autoscaling of gateway pods](#)
Enable autoscaling of gateway pods to ensure high availability of DataPower pods.
- [Dynamically re-registering and reconfiguring a Gateway service in a Kubernetes deployment](#)
In API Connect, Gateway services do not persist their configuration settings by default. Instead, the master configuration is stored on the Management server and the *Dynamic Reregistration and Reconfiguration* (DRR) mechanism resynchronizes configuration data when needed. The DRR process is used when proper High Availability/Disaster Recovery (HA/DR) is not configured, or if a manual resynchronization is required.

Customizing a DataPower deployment

Update the `GatewayCluster` CR to configure `DataPowerService` APIs to customize your DataPower deployment.

You can set the following properties in the CR. Note that each `GatewayCluster` CR property has a corresponding `DataPowerService` API. Refer to the DataPower Operator documentation link for each property:

Table 1. Mapping of `GatewayCluster` properties to `DataPowerService` properties

GatewayCluster property	DataPowerService property	DataPowerService API Docs
<code>AdditionalDomainConfig</code>	<code>domains</code>	domains For domain configuration, see also: <ul style="list-style-type: none">• Guides: Domain Configuration• Features: Configuration Management
<code>AdditionalExtraExe</code>	<code>extraExe</code>	extraExe
<code>AdditionalInitCmds</code>	<code>initCmds</code>	initCmds
<code>lifecycle</code>	<code>lifecycle</code>	lifecycle
<code>healthcheck</code>	<code>healthcheck</code>	healthcheck
<code>readinessTimeoutSeconds</code>	<code>readinessTimeoutSeconds</code>	readinessTimeoutSeconds
<code>hostAliases</code>	<code>hostAliases</code>	hostAliases

Examples of how to add the DataPower service properties to a `GatewayCluster` CR, in an API Connect deployment on native Kubernetes:

- Adding `additionalDomainConfig`:

```
spec:
  additionalDomainConfig:
  - name: "default"
    certs:
  - certType: "usrcerts"
    secret: "default-cert"
  - certType: "sharedcerts"
    secret: "shared-cert"
  dpApp:
    config:
  - "default-config"
    local:
  - "default-local"
```

- Extending the `apiconnect` domain:

```
spec:
  additionalDomainConfig:
  - name: "apiconnect"
    dpApp:
      config:
  - "custom-apiconnect-config"
      local:
  - "custom-apiconnect-local"
```

Overriding the default DataPower image, version, and license

Update the `GatewayCluster` CR to override the image, version, and license information for the Gateway subsystem.

Procedure

To override these settings in the CR, add a `spec.dataPowerOverride` section as shown in the following example:

```
spec:
  dataPowerOverride:
    image: customregistry.com/custom-image-datapower:10.5.0.3
    version: 10.5.0.3
    license: X-XXXX-XXXXXX
```

Enabling autoscaling of gateway pods

Enable autoscaling of gateway pods to ensure high availability of DataPower pods.

Autoscaling enables the gateway deployment to scale dynamically, either vertically or horizontally.

It is not possible to use both horizontal and vertical pod scaling simultaneously. You must configure either **HPA** (for horizontal scaling) or **VPA** (for vertical scaling) as scaling method.

- To enable HPA, add the following fields to the CR:
In OpenShift or Cloud Pak for Integration, add the fields to the `spec.gateway` section of the API Connect Cluster CR.

```
podAutoScaling:
  hpa:
    maxReplicas: 5
    minReplicas: 3
    targetCPUUtilizationPercentage: 90
    targetMemoryUtilizationPercentage: 80
  method: HPA
```

- To enable VPA, add the following fields to the CR:
In OpenShift or Cloud Pak for Integration, add the fields to the `spec.gateway` section of the API Connect Cluster CR.

```
podAutoScaling:
  method: VPA
  vpa:
    maxAllowedCPU: 4
    maxAllowedMemory: 16G
```

Note that the values shown in the CR entries are just examples.

Table 1. Autoscaling configuration settings

Setting	Description
<code>method</code>	Specifies either Horizontal Pod Autoscaling or Vertical Pod Autoscaling Value must be either HPA or VPA . To disable autoscaling, leave <code>method</code> blank or omit the <code>podAutoScaling</code> section from the CR.
<code>maxReplicas</code>	HPA. Maximum pods that can be scaled up to by the horizontal pod autoscaler. Must be a value greater than <code>minReplicas</code> .
<code>minReplicas</code>	HPA. Minimum pods that can be scaled down to by the horizontal pod autoscaler.
<code>targetCPUUtilizationPercentage</code>	HPA. Percent threshold on average CPU usage by pods for triggering horizontal scaling.
<code>targetMemoryUtilizationPercentage</code>	HPA. Percent threshold on average memory usage by pods for triggering horizontal scaling.
<code>maxAllowedCPU</code>	VPA. Maximum value for CPU resource that can be assigned to a DataPower container when scaling vertically. Minimum value is set from <code>DataPowerRequestsSpec.ResourceCPU</code>
<code>maxAllowedMemory</code>	VPA. Maximum amount of memory that can be assigned to a DataPower container when scaling vertically. Minimum value is set from <code>DataPowerRequestsSpec.ResourceMemory</code>

Note:

DataPower Gateway supports long-lived connections such as GraphQL subscriptions or other websockets connections. These long-lived connections might not be preserved when pods are scaled down based on defined CPU or memory thresholds. Workloads with long-lived connections are more vulnerable to failed API transactions when auto-scaling is occurring.

To learn more about HPA and VPA autoscaling, see

- <https://ibm.github.io/datapower-operator-doc/features/pod-auto-scaling/>
- <https://ibm.github.io/datapower-operator-doc/apis/datapowerservice/v1beta3#podautoscaling>

Dynamically re-registering and reconfiguring a Gateway service in a Kubernetes deployment

In API Connect, Gateway services do not persist their configuration settings by default. Instead, the master configuration is stored on the Management server and the *Dynamic Reregistration and Reconfiguration* (DRR) mechanism resynchronizes configuration data when needed. The DRR process is used when proper High Availability/Disaster Recovery (HA/DR) is not configured, or if a manual resynchronization is required.

If a Gateway service is not configured properly for resiliency and is restarted, the gateways in the Gateway service will lose the configuration from the Management server. Configuration data from the Management server is maintained on the gateway service according to the gateway peering configuration on the gateways.

Preventing the loss of configuration data

For high availability in production environments, use a minimum of three gateways in the Gateway service.

Forcing re-population of the configuration data

To repopulate the configuration data, complete the steps in the following section to trigger an outage-less DRR.

Triggering an outage-less DRR

To force a Dynamic Reregistration and Reconfiguration across a peer group from within the gateway service without requiring a restart, complete the following steps using the CLI. This process flushes the primary peering object configured for the apic-gw-service. This task can only be performed with CLI, and requires the default admin ID for running the `diag` command.

1. Access the gateway service:

- For a virtual gateway, run the following command to access the DataPower CLI:

```
kubectl attach -it <pod_name>
```

- For a physical gateway, use `ssh` to connect to the gateway service.

2. Switch to the API Connect application:

```
switch <APIC_APP_DOMAIN>
```

3. Show the current gateway-peering-status:

```
show gateway-peering-status
```

4. If the server that you connected to is not primary for the apic-gw-service gateway-peering object, force this server to become primary:

```
config; gateway-peering-switch-primary <gateway-peering-name-for-apic-gw-service>;
```

5. Flush the apic-gw-service gateway-peering object:

```
diag; gateway-peering-flush <gateway-peering-name-for-apic-gw-service>; exit
```

When prompted, confirm the operation.

6. Disable and then re-enable the apic-gw-service object for every member of the gateway service:

```
config; apic-gw-service; admin-state disabled; exit
apic-gw-service; admin-state enabled; exit; exit
```

7. Confirm that the apic-gw-service was flushed and is now waiting for gateway service registration:

Look in the debug log target for the apic-gw-service, located in `logtemp:///` and check for a message similar to the following example:

```
20200618T232339.332Z [0x88e000d7][apic-gw-service][notice]
apic-gw-service(default): tid(2653): Waiting for gateway service registration.
```

The DRR will be triggered by the next arrival of a webhook event. The Management server sends a heartbeat to the gateway at five-minute intervals, prompting the gateway to check whether it has lost its configuration and if so, trigger a DRR. In addition, if the Management server has previously marked a Catalog or cloud as being unavailable for a particular gateway service, a successful heartbeat triggers synchronization for that Catalog or cloud on that gateway service.

8. If you attached to a virtual gateway, you can detach by pressing Ctrl+P and then pressing Ctrl+Q.

9. (Optional) Force a BAU webhook event to be sent from API Manager to trigger the DRR:

If you do not want to wait for the Management server to send a heartbeat, you can trigger the DRR manually by completing the following steps:

- a. Open the API Manager.
- b. Open the Catalog.
- c. Click Settings, > Edit.
- d. Update the Summary field with some text so that it is modified.
- e. Click Save.

When the event is received, the DRR is initiated.

Backing up and restoring

You can backup and restore API Connect subsystems.

API Connect Version 10 uses Kubernetes operators to back up and restore databases.

- Backups are for recovery of the API Connect subsystems in the same environment that the backups were taken. If you want to move your API Connect deployment to a different environment and change the form factor or modify your API Connect endpoints, then see [Migrating from v10 to v10 on a different form factor](#).
- Manual backups
For the management database, the backup configuration is specified in a Management CR (custom resource). You create the backup CR manually. Once the backup CR is created, the `apiconnect` operator performs backup and updates the status into the backup CR.
- Scheduled backups
For scheduled backups, you can add a backup schedule to the backup configuration details in the management CR. The `apiconnect` operator will then trigger and perform backups as specified in the custom `cron` schedule. A backup CR is created for every individual backup invocation as described in the cron schedule. You can view each backup CR to determine the status of each backup.
- Restore
For restore, you update the backup Management CR. You then create a restore CR with the backup name in it. You can list the backup CRs and determine the backup name by looking at individual backup CR. The `apiconnect` operator performs the restore.
- You must backup both the Management and Portal subsystems at the same time, to ensure synchronicity across the services.
- If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. The backups of the Management and Portal must be taken at the same time to ensure that the Portal sites are consistent with Management database.
- When restoring, the Gateway and all deployed subsystems (Management, Analytics, and Developer Portal) must be at the same version level.

Note: For information about how to maintain a two data center deployment, see [Maintaining a two data center deployment](#).

- **Backups on Kubernetes**

Use the procedures in this section to backup and restore API Connect on generic Kubernetes distributions.

- [Backups on OpenShift](#)
You can backup and restore API Connect subsystems on OpenShift and Cloud Pak for Integration.

Backups on Kubernetes

Use the procedures in this section to backup and restore API Connect on generic Kubernetes distributions.

- [Backing up and restoring the management subsystem](#)
You can use the following instructions to back up and restore your Management subsystem on API Connect.
- [Backing up and restoring the Developer Portal in a Kubernetes environment](#)
You can back up and restore your Developer Portal service in your Kubernetes environment.
- [Backing up and restoring the analytics database](#)
The IBM® API Connect Analytics database can be backed up and restored from an S3 repository. S3 compatible object storage is required; for example, IBM Cloud Object Storage.

Backing up and restoring the management subsystem

You can use the following instructions to back up and restore your Management subsystem on API Connect.

- To complete a backup and restore of your Management subsystem, you must have the **custom resource** configured with the **databaseBackup** subsection in the management subsystem custom resource.
- Backups can be scheduled or generated manually (on-demand). You can then list what backups are available and then use one to restore your Management subsystem if required:

```
kubect1 get mgmtb -n <namespace>
```

- Postgres replica pods depend on a successful completed backup. If backup configuration is not correct, replica pods won't come up.
- Custom resource (CR) templates are provided in the **helper_files** archive to install each API Connect subsystem. You decompressed these files into an installation folder of your choosing in [Obtaining product files](#).
- For points to consider when backing up and restoring a two data center disaster recovery deployment on Kubernetes, see [Backup and restore requirements for a two data center deployment](#).
- Restoring a backup restores the registration credentials (client_ID, client_secret) that were in use at the time that the selected backup was created. For information on the registration credentials, see [Changing the registration client id and client secret for applications](#).

Note: You must backup both the Management and Portal subsystems at the same time, to ensure synchronicity across the services.

Types of backups supported

- Cloud-based S3 storage
 - IBM Cloud Object Storage
 - AWS S3
 - **v10.0.2.0 or greater**: Any custom S3, such as minIO.
- SFTP
- Local storage backups
- [Configuring backup settings for fresh install of the Management subsystem](#)
You can configure backups for your Management subsystem in your Kubernetes environment.
- [Reconfiguring or adding backup settings after installation of the management subsystem](#)
You can reconfigure the backup settings after installation of the management subsystem.
- [Generating a manual backup of the management subsystem](#)
You can generate a manual (non-scheduled) backup of the management subsystem
- [Restoring the management subsystem \(v10.0.1.1 or later\)](#)
You can restore your Management subsystem in your Kubernetes environment.
- [Troubleshooting management subsystem backups](#)
You can troubleshoot failures of the management subsystem backups.

Configuring backup settings for fresh install of the Management subsystem

You can configure backups for your Management subsystem in your Kubernetes environment.

Use the instructions for your backup type:

- [Configuring S3 backup settings for fresh install of the Management subsystem](#)
You can configure backups for your management subsystem in your Kubernetes environment.
- [Configuring SFTP backup settings for fresh install of the Management subsystem](#)
You can configure backups for your Management subsystem in your Kubernetes environment.
- [Configuring local backup settings for fresh install of the Management subsystem](#)
You can configure backups for your Management subsystem in your Kubernetes environment.

Configuring S3 backup settings for fresh install of the Management subsystem

You can configure backups for your management subsystem in your Kubernetes environment.

Before you begin

Review [Backing up and restoring the management subsystem](#).

If you have a two data center disaster recovery deployment, before you configure S3 backup settings you must first disable 2DCDR on both sites. Follow these steps:

1. Remove the `multiSiteHA` section from the `spec` section of the management CRs on both data centers. Take note of which data center is the active and which is the warm-standby. Keep a copy of the `multiSiteHA` sections that you remove.
2. Configure your S3 backup settings. You must specify different backup locations for each data center, see [Backup and restore requirements for a two data center deployment](#).
3. Add the `multiSiteHA` sections back to the management CRs in both data centers, ensuring that your original active is set to be the active again.

About this task

If you haven't already, configure your management subsystem `custom resource` with the `databaseBackup` subsection.

Note: `S3Provider` supports `custom` S3 solutions in 10.0.2.0 and greater. The support includes new backup parameters `backupCerts` and `backups3URISyle`.

Important: For S3 backups of the management subsystem, do not use retention features provided by Cloud-based S3 storage providers. Use of such features can result in periodic deletion of archived backups, which can cause backup corruptions which can cause database restores to fail.

Warning:

When configuring a new management subsystem for S3 backup, the content of the S3 bucket path must be empty. If you want to configure a management subsystem to use the content of an S3 bucket path already used by another management subsystem, you have two options:

1. Shutdown the postgres database on the existing management subsystem that points to the S3 bucket path before you start the new management subsystem.
2. In a disaster recovery scenario where you want to restore from the backup of an existing installation: Create a new S3 path and use S3 tools to copy the backup content from the existing management subsystem S3 bucket path, to the new bucket path.

Procedure

1. Create a backup secret.

The backup secret is a Kubernetes secret that contains your username and password for your S3 backup database.

For examples of how to generate the appropriate access key and secret for two of them, see:

- IBM (Cloud Object Storage): [Service credentials](#).
- AWS: [Managing access keys](#).
- Custom: See documentation for your S3 provider. For example, minIO.

The secret can be created with the following command:

```
$ kubectl create secret generic mgmt-backup-secret --from-literal=key='<YOUR ACCESS KEY>' \
--from-literal=keysecret='<YOUR ACCESS KEY SECRET>' -n <namespace_of_management_subsystem>
```

2. Ensure that your Management subsystem `custom resource` is configured with the `databaseBackup` subsection.

For example:

```
databaseBackup:
  protocol: objstore
  s3provider: ibm
  host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
  path: apic-backup
  credentials: mgmt-backup-secret
  backupCerts: <custom-s3-server-CA-cert>
  backups3URISyle: host(default)/path
  schedule: "0 3 * * *"
  repoRetentionFull: 2
```

Note: `backupCerts` and `backups3URISyle` are used for custom s3 only, on **v10.0.2.0 or greater**.

Table 1. Backup configuration settings

Setting	Description
<code>protocol</code>	The type of the backup. For s3 storage: <code>objstore</code> . When the parameter is not set, the backup type defaults to local storage backup.
<code>s3provider</code>	Name of the S3 provider to use. You must specify one of the supported values of <code>aws</code> , <code>ibm</code> , or <code>custom</code> . Support for <code>custom</code> is available in v10.0.2.0 or greater . Note: The public certificate on the S3 storage provider must be signed by a known certificate authority that is trusted by API Connect. Use of an untrusted authority can cause the following error during backup upload: <code>x509: certificate signed by unknown authority</code> .
<code>host</code>	For <code>objstore</code> type backup, specify S3 endpoint with the corresponding S3 region in the format <code><S3endpoint>/<S3region></code>
<code>path</code>	The path to the location of the backup: For <code>objstore</code> backups, the name of your S3 bucket to store backup data. For example, <code>bucket-name/folder</code> . The use of subdirectories in the bucket name is not supported. Note: When your deployment includes a management subsystem in two different clusters (with active databases), the two management subsystems cannot use the same s3 bucket name in the database backup configurations. Each management subsystem must use a unique s3 bucket name. Ensure that <code>bucket-name/folder</code> is empty. If the folder was previously used for backups, the folder will not be empty, and stanza create might encounter an error. Explanation: Once a folder is used, <code>archive.info</code> and <code>backup.info</code> files are created. During stanza creation, the process compares the database version and database system id between the two info files and the current Postgres database cluster. Stanza creation fails if there is a mismatch. To prevent this possibility, remove all files from the folder before configuring.

Setting	Description
credentials	For objstore type backups, the name of the Kubernetes secret containing your S3 Access Key / Key Secret
backupCerts	<p>v10.0.2.0 or greater</p> <p>Custom certificate. Used only when s3Provider is set to custom. Most of the custom S3 providers are created based on custom CA certificate.</p> <p>This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be ca.crt and value should be base64 encoded value of CA certificate.</p> <p>The API Connect management subsystem validates the custom S3 certificate against the customer-provided CA certificate.</p> <p>Sample bash script to create the Kubernetes secret:</p> <pre>cat >customs3ca.yaml <<EOF apiVersion: v1 data: ca.crt: \$(base64 <path-to-ca-certificate> tr -d '\n') kind: Secret metadata: name: custom-server-ca type: generic EOF</pre> <p>kubectl apply -f customs3ca.yaml -n <namespace></p>
backups3URIS tyle	<p>v10.0.2.0 or greater</p> <p>Valid values: host and path.</p> <p>Only allowed if s3Provider is set to custom.</p> <p>Some custom S3 providers require URI style to be set to path. For example, minIO supports both host and path style setup. You can create a minIO S3 server to only accept path style client communications.</p> <p>Important: Contact your custom S3 administrator before configuring this field. If not properly configured, upstream custom S3 can reject connections from the client, such as API Connect management subsystem.</p>
schedule	<p>Cron like schedule for performing automatic backups. The format for the schedule is:</p> <ul style="list-style-type: none"> • ***** • ----- • • +----- day of week (0 - 6) (Sunday=0) • +----- month (1 - 12) • +----- day of month (1 - 31) • ----- hour (0 - 23) • +----- min (0 - 59) <p>The timezone for backups is that of the node on which the postgres-operator pod is scheduled.</p> <p>There is no default backup schedule set. Be sure to set your backup schedule.</p> <p>All scheduled Management subsystem backups are of type full only.</p>
repoRetentio nFull	<p>v10.0.3.0 or later - The number of full S3 backups to retain. When the next full backup successfully completes, and the specified number of retained backups is reached, the oldest full backup is deleted. All incremental backups and archives associated with the oldest full backup also expire. Incremental backups are not counted for this setting.</p> <p>Applies to both manual backups and scheduled backups.</p> <p>Minimum value: 1</p> <p>Maximum value: 9999999</p> <p>Default: none</p>

3. Verify that the configuration deploys successfully.

When you configure Management subsystem backups, the operator runs a **stanza-create** job. This job creates the stanza (Postgres cluster configuration) on the upstream S3 server, which is used for backup and archive procedures. The job also brings up the necessary pod.

Check the status of the **stanza-create** job:

```
kubectl get jobs -n <namespace> | grep stanza
```

- On success:

- The **stanza-create** job status is 1/1:

```
kubectl get jobs | grep stanza
m1-b722e361-postgres-stanza-create 1/1 5s 127m
```

- The **stanza-create** job shows **status: "True"** and **type: Complete**:

```
kubectl get job m1-b722e361-postgres-stanza-create -o yaml
```

```
status:
  completionTime: "2021-04-13T18:38:50Z"
  conditions:
  - lastProbeTime: "2021-04-13T18:38:50Z"
    lastTransitionTime: "2021-04-13T18:38:50Z"
    status: "True"
    type: Complete
  startTime: "2021-04-13T18:38:45Z"
  succeeded: 1
```

- The `stanza-create` pod is in `Completed` state:

```
kubectl get pods | grep stanza
m1-b722e361-postgres-stanza-create-q4fvp          0/1    Completed    0          127m
```

- Exec into the pod:

```
kubectl exec -it <backrest-shared-repo-pod> -- bash
```

- The `pgbackrest` command returns the S3 contents in valid JSON:

```
pgbackrest info --output json --repol-type s3
{"backup":{"held":false},"message":"ok"}} [pgbackrest@m1-b722e361-postgres-backrest-shared-repo-69cdfdfd5-rs882 /
```

- On failure, see [Troubleshooting stanza-create job for S3 backup configuration](#).

Configuring SFTP backup settings for fresh install of the Management subsystem

You can configure backups for your Management subsystem in your Kubernetes environment.

Before you begin

Review [Backing up and restoring the management subsystem](#).

About this task

If you haven't already, configure your Management subsystem `custom resource` with the `databaseBackup` subsection.

Procedure

1. Create a backup secret.

The backup secret is a Kubernetes secret that contains your credentials for accessing the SFTP backup database. Supported credentials types:

- Username and password (v10.0.2.0 or later).
- Username and SSH-key (v10.0.3.0 or later). Only OpenSSH keys are supported.¹

Use one of the following commands to create the secret:

- Username and password credentials

```
$ kubectl create secret generic mgmt-backup-secret --from-literal=username='<YOUR USERNAME>' --from-literal=password='<YOUR PASSWORD>' -n <namespace-of-mgmt-subsystem>
```

- Version 10.0.3.0 or later:** Username and SSH-key credentials:

```
$ kubectl create secret generic mgmt-backup-secret --from-literal=username='<YOUR USERNAME>' --from-file=ssh-privatekey='<YOUR PRIVATEKEY FILE>' -n <namespace-of-mgmt-subsystem>
```

Note: If the username contains the backslash character, insert an extra backslash to escape it. For example, if the username is:

```
backup\svc-apic
```

set this as:

```
--from-literal=username='backup\\svc-apic'
```

2. Ensure that your Management subsystem `custom resource` is configured with the `databaseBackup` subsection.

For example:

```
databaseBackup:
  protocol: sftp
  host: <SFTP-host-name>
  port: <SFTP-port>
  path: apic-backup
  retries: 0
  credentials: mgmt-backup-secret
  schedule: "0 3 * * *"
```

Table 1. Backup configuration settings

Setting	Description
<code>protocol</code>	The type of the backup. For SFTP storage: <code>sftp</code> .
<code>host</code>	The backups host. For <code>sftp</code> type, the SFTP server hostname
<code>port</code>	The SFTP server port. Optional. Default: <code>22</code> .
<code>path</code>	The path to the location of the backup. For <code>sftp</code> type, the full absolute path of the folder on the SFTP server, beginning with <code>/</code> .
<code>retries</code>	The number of times the <code>ibm-apiconnect</code> Operator attempts backups in the event of a failed SFTP backup. Default value: <code>0</code> .
<code>credentials</code>	Name of the Kubernetes secret containing your SFTP Username/password. For Version 10.0.3.0 or later, the Kubernetes secret can contain your SFTP Username and SSH key.

Setting	Description
<code>schedule</code>	<p>Cron like schedule for performing automatic backups. The format for the schedule is:</p> <ul style="list-style-type: none"> • <code>*****</code> • <code>-----</code> • <code> </code> • <code> +-----</code> day of week (0 - 6) (Sunday=0) • <code> +-----</code> month (1 - 12) • <code> +-----</code> day of month (1 - 31) • <code> +-----</code> hour (0 - 23) • <code>+-----</code> min (0 - 59) <p>The timezone for backups is that of the node on which the postgres-operator pod is scheduled.</p> <p>There is no default backup schedule set. Be sure to set your backup schedule.</p> <p>All scheduled Management subsystem backups are of type <code>full</code> only.</p>

¹ PuTTY style keys can be converted to OpenSSH by using the PuTTY Key Generator (PuTTYgen) application; see <https://www.puttygen.com/>.

Configuring local backup settings for fresh install of the Management subsystem

You can configure backups for your Management subsystem in your Kubernetes environment.

Before you begin

Review [Backing up and restoring the management subsystem](#).

About this task

If you haven't already, configure your Management subsystem `custom resource`.

Procedure

Configure your Management subsystem `custom resource` with the `databaseBackup` subsection. Specify `schedule` and `repoRetentionFull`, the default settings are 01:00 UTC and 14 days.

```
databaseBackup:
  schedule: "0 1 * * *"
  repoRetentionFull: 14
```

Table 1. Backup configuration settings

Setting	Description
<code>schedule</code>	<p>Cron like schedule for performing automatic backups. The format for the schedule is:</p> <ul style="list-style-type: none"> • <code>*****</code> • <code>-----</code> • <code> </code> • <code> +-----</code> day of week (0 - 6) (Sunday=0) • <code> +-----</code> month (1 - 12) • <code> +-----</code> day of month (1 - 31) • <code> +-----</code> hour (0 - 23) • <code>+-----</code> min (0 - 59) <p>The timezone for backups is that of the node on which the postgres-operator pod is scheduled.</p> <p>The default is daily backups at 01:00 UTC.</p> <p>All scheduled Management subsystem backups are of type <code>full</code> only.</p>
<code>repoRetentionFull</code>	<p>- The number of full local backups to retain. When the next full backup successfully completes, and the specified number of retained backups is reached, the oldest full backup is deleted. All incremental backups and archives associated with the oldest full backup also expire. Incremental backups are not counted for this setting.</p> <p>Applies to both manual backups and scheduled backups.</p> <p>Minimum value: 1</p> <p>Maximum value: <code>9999999</code></p> <p>Default: 14</p>

Reconfiguring or adding backup settings after installation of the management subsystem

You can reconfigure the backup settings after installation of the management subsystem.

Procedure

1. Make sure that the management cluster status is **Running** and the READY condition displays the same value before and after the "/".

For example:

```
k get mgmt
NAME   READY   STATUS    VERSION      RECONCILED VERSION  AGE
mgmt   16/16   Running   10.0.2.0     10.0.2.0-xxxx      19h
```

2. Create backup secret as described in [Configuring backup settings for fresh install of the Management subsystem](#).

3. Complete the following steps, based on your storage type:

- S3 storage

In 10.0.1.0 and greater, you can change or add the **DatabaseBackup** s3 configuration via the Management Cluster CR after fresh install. However, doing so requires a reconfiguration of the database and a brief downtime. The operator restarts the database with the new backup configurations as soon as the new configuration in the management CR is saved.

If you have a two data center disaster recovery deployment, before you configure S3 backup settings you must first disable 2DCDR on both sites. Follow these steps:

- Remove the **multiSiteHA** section from the **spec** section of the management CRs on both data centers. Take note of which data center is the active and which is the warm-standby. Keep a copy of the **multiSiteHA** sections that you remove.
- Configure your S3 backup settings. You must specify different backup locations for each data center, see [Backup and restore requirements for a two data center deployment](#).
- Add the **multiSiteHA** sections back to the management CRs in both data centers, ensuring that your original active is set to be the active again.

Note:

If you have an APICoordinateCluster custom resource installed, do not modify the ManagementCluster CR. Instead, modify the APICoordinateCluster CR.

You can verify if you have an **apiconnectcluster** CR with the command

```
$ kubectl get apiconnectcluster -n <your-namespace>
```

Configuration of s3 backups uses a new parameter **restartDB** is required when changing the backup configuration. This parameter is in addition to the parameters described in [Configuring backup settings for fresh install of the Management subsystem](#).

Example CR:

```
databaseBackup:
  protocol: objstore
  s3provider: ibm
  host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
  path: apic-backup
  credentials: mgmt-backup-secret
  schedule: "0 3 * * *"
  restartDB:
    accept: true
```

Table 1. Additional parameter for S3 backup type reconfiguration

Parameter	Description
restartDB: accept: <value>	Required when changing the backup configuration. The parameter accept is a boolean: <value> can be true or false . When accept is true , and a new configuration for management CR is saved, the operator: <ul style="list-style-type: none">• Stops the database and any dependent services• Applies the change• Restarts the stopped components. Note: restartDB is not required for initial deployments, only for modifications after the cluster is deployed.

- SFTP

You can change/add the **DatabaseBackup** SFTP configuration via the Management Cluster CR after fresh install. This process does not involve any downtime. The same backup settings are used for reconfiguring as for initial configuration. For complete description of backup settings, see [Configuring backup settings for fresh install of the Management subsystem](#).

Example CR:

```
protocol: sftp
host: <SFTP-hostname>
port: <SFTP-port>
path: apic-backup
retries: 0
credentials: mgmt-backup-secret
schedule: "0 3 * * *"
```

Generating a manual backup of the management subsystem

You can generate a manual (non-scheduled) backup of the management subsystem

Procedure

1. Make sure that the management cluster status is **Running** and the READY condition displays the same value before and after the "/".

For example:

```
kubectl get mgmt
NAME   READY   STATUS    VERSION      RECONCILED VERSION  AGE
mgmt   16/16   Running   10.0.2.0     10.0.2.0-xxxx      19h
```

Note: Be sure **READY** status is 16/16. If you have just configured or reconfigured the backup settings, there is a period of time when the postgres pods are taken offline and brought back up with the new configuration. During this time, the `kubectl get mgmt` command will show 11/11 instead of 16/16. If you start a backup before the Management cluster is on 16/16, the backup attempt might stall indefinitely.

- Decide which of the two types of manual backups you want to do:
 - full**: backs up the entire Management Subsystem database.
 - incr**: backs up any data that has not been backed up since the last **full** or **incr** backup.
- You generate a backup manually by creating a `managementBackup` custom resource which is detected by the `ibm-apiconnect` operator that then triggers the backup process.

You can use `mgmtbackup_cr.yaml` in the `helper_files` directory as an example:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementBackup
metadata:
  generateName: management-
spec:
  type: full
  crType: create
  clusterName: management
```

Table 1 lists the CR settings for a manual backup.

Table 1. CR settings for manual backup

Setting	Description
<code>type</code>	The type of backup to be done. Valid arguments are: <ul style="list-style-type: none"> full Backs up the entire Management subsystem database incr Backs up any data that has not been backed up since the last full or incr backup.
<code>crType</code>	Use create to trigger a backup process.
<code>clusterName</code>	The name of the target ManagementCluster you want to backup.

- Create the `ManagementBackup` custom resource in the namespace of your Management Subsystem to trigger the backup process:

```
$ kubectl create -f mgmtbackup_cr.yaml -n <namespace-of-mgmt-subsystem>
```

- (Optional): You can list current backups by using the command:

```
$ kubectl get managementbackup -n <namespace-of-mgmt-subsystem> --sort-by=.metadata.creationTimestamp
```

Here is an example of some possible output:

NAME	STATUS	ID	CLUSTER	TYPE	CR TYPE	AGE
m1-e46bbac4	Ready	20201023-183005F	m1	full	record	66m
mgmt-backup-dnvxw	Ready	20201023-193137F	m1	full	create	9m37s

- (Optional): You can get a detailed view of a specific backup by using the command:

```
$ kubectl get managementbackup <bup-name> -n <namespace-of-mgmt-subsystem> -o yaml
```

Here is an example of some possible output:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementBackup
metadata:
  creationTimestamp: "2020-10-23T18:34:11Z"
  generation: 1
  labels:
    app.kubernetes.io/instance: m1
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: m1-e46bbac4
  name: m1-e46bbac4
  namespace: default
  resourceVersion: "6869"
  selfLink: /apis/management.apiconnect.ibm.com/v1beta1/namespaces/default/managementbackups/m1-e46bbac4
  uid: b7b6be9c-82d4-4737-984f-58584a3fe582
spec:
  clusterName: m1
  crType: create
  type: full
status:
  clusterName: m1-fbaa5be2-postgres
  conditions:
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupPending
    status: "False"
    type: Pending
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    message: Record of management backup complete
    reason: BackupComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupStatusUnknown
```

```

status: "False"
type: Warning
crType: ""
id: 20201023-183005F
info:
  id: 20201023-183005F
  range: 2020-10-23 18:30:05 +0000 UTC - 2020-10-23 18:31:22 +0000 UTC
  rangeEnd: 1603477882
  rangeStart: 1603477805
  size: 32.6 MB
  totalsize: 32.6 MB
  type: full
phase: Ready
state: ""
subsysName: ml

```

Restoring the management subsystem (v10.0.1.1 or later)

You can restore your Management subsystem in your Kubernetes environment.

Before you begin

You must have backups of the management subsystem in order to restore. See [Configuring backup settings for fresh install of the Management subsystem](#) and [Generating a manual backup of the management subsystem](#).

About this task

To trigger a restore, you will create a **ManagementRestore** custom resource. When the **ibm-apiconnect** operator detects the new CR, it starts the restore process.

- Ensure that the backup method that you used is complete before you attempt a restore. The backup data that is stored in the remote server is used for restoring the Management subsystem data back to a previous state. Ensure that these backups are on your remote server before you attempt a restore, and that the backup ID in the selected backup exists in remote storage.
- Restoring a backup restores the registration credentials (client_ID, client_secret) that were in use at the time that the selected backup was created. For information on the registration credentials, see [Changing the registration client id and client secret for applications](#).
- If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. The backups of the Management and Portal must be taken at the same time to ensure that the Portal sites are consistent with Management database.

Note: There is a known issue where sometimes a restore from the Management subsystem's SFTP backup succeeds but the data is not restored. If this happens, run the restore again.

Procedure

1. Obtain a list of the available backups, and decide which one to restore your Management subsystem:

```
kubectl get mgmtb -n <your-namespace>
```

For example:

```

$ kubectl get mgmtb
NAME                                STATUS  ID                                CLUSTER  TYPE  CR TYPE  AGE
management-3aa3bebf                 Ready  20200929-152150F                 management  full  record  10h
management-3c4ca8df                 Ready  20200929-115520F                 management  full  record  20h
management-5tbxj                    Ready  20200929-224349F                 management  full  create   17h
management-f10af337                 Ready  20200925-133703F                 management  full  record  5d2h
management-jtlcd                    Ready  20200929-224349F                 management  full  create   25h
management-zxjtz                    Ready  20200929-224349F                 management  full  create   28h

```

Attention: You can restore the Management subsystem from any backup where the STATUS is Ready; however, you should avoid restoring to the initial system backup. During installation, the Management database is used for an initial system backup before certain database schema jobs are complete. Restoring to this backup will result in an unstable system.

You will use one of the values in the **NAME** column.

2. Configure **mgmtrestore_cr.yaml**. Use the copy in the **helper_files** directory as an example:

```

apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementRestore
metadata:
  generateName: management-
spec:
  backupName: management-3aa3bebf

```

backupName: is the name of the backup custom resource you want to restore to. Note that the sample uses example output from the previous step, to specify the backup named **management-3aa3bebf**:

```
backupName: management-3aa3bebf
```

3. Create the **ManagementRestore** CR in the namespace of the Management Subsystem to trigger the restore process:

```
$ kubectl create -f mgmtrestore_cr.yaml -n <namespace-of-mgmt-susbsystem>
```

4. You can list current restores using the following command:

```
$ kubectl get managementrestore -n <namespace-of-mgmt-subsytem> --sort-by=.metadata.creationTimestamp
```

Here is an example of some possible output:

- SFTP:


NAME	STATUS	BACKUP	CLUSTER	MESSAGE
PITR	AGE			
mgmt-restore-ptx6d	Complete	mgmt-backup-w4h8c	m1	Restore process completed (DB Restore + DRR job)
6m31s				

- S3

NAME	STATUS	BACKUP	CLUSTER	MESSAGE
PITR	AGE			
mgmt-restore-hr9zx	Complete	mgmt-backup-xnjkl	m1	Restore process completed (DB Restore + DRR job)
2020-11-23 23:08:32+00	9m8s			

Troubleshooting management subsystem backups

You can troubleshoot failures of the management subsystem backups.

Note: In the  Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
See:

- [Troubleshooting resiliency issues](#)
Resolve resiliency issues with API Connect management system backups.
- [Troubleshooting stanza-create job for S3 backup configuration](#)
Troubleshoot configuration of s3 backups when the stanza-create job fails.
- [Restore failure with bootstrap pod error](#)
Postgres bootstrap pod reports an error and restore does not progress.

Troubleshooting resiliency issues

Resolve resiliency issues with API Connect management system backups.

pgbackrest-shared-repo resiliency problems

If the Kubernetes node hosting pgbackrest-shared-repo pod is down and the PVC attached to the pod has strict zone requirements (for example, in AWS or other clouds) or if the storage class is set to `local-storage`, then pgbackrest-shared-repo pod will not get rescheduled to another Kubernetes node.

As a result, there is a single point of failure and the following conditions might occur:

- Backups of the management database fails
- Disk space fills with accumulated Postgres wal (Write-Ahead Logging) files

To avoid this problem, monitor the disk usage: [Monitoring Postgres disk usage](#).

When the Kubernetes node comes back up, the pod is scheduled and all the processes should resume properly.

Troubleshooting stanza-create job for S3 backup configuration

Troubleshoot configuration of s3 backups when the stanza-create job fails.

When you configure Management subsystem backups for s3 providers, the operator runs a `stanza-create` job. This job creates the stanza (Postgres cluster configuration) on the upstream S3 server, which is used for backup and archive procedures. The job also brings up the necessary pod.

Check the status of the `stanza-create` job:

```
kubectl get jobs -n <namespace> | grep stanza
```

On failure:

- `stanza-create` pods fail, and job status is 0/1:

```
kubectl get jobs | grep stanza
m1-3bfe12ac-postgres-stanza-create      0/1      32m      32m
```

- Listing all pods brought up by `stanza-create` job shows multiple pods because a job tries `backoffLimit` times to complete the job. By default, there will be 7 pods in error state:

```
kubectl get pods | grep stanza
m1-3bfe12ac-postgres-stanza-create-726z4 0/1      Error    0      11m
m1-3bfe12ac-postgres-stanza-create-9vq4n 0/1      Error    0      22m
m1-3bfe12ac-postgres-stanza-create-gbtgb 0/1      Error    0      30m
m1-3bfe12ac-postgres-stanza-create-g15g5 0/1      Error    0      17m
m1-3bfe12ac-postgres-stanza-create-q8qhw 0/1      Error    0      26m
```

m1-3bfe12ac-postgres-stanza-create-t84zs	0/1	Error	0	8m54s
m1-3bfe12ac-postgres-stanza-create-z8tfs	0/1	Error	0	12m

Note: In some cases, the pods are cleaned by Kubernetes and the logs of these pods can be lost.

- Pod log errors show a reason for the failure. For example, when a hostname is not valid:

```
kubectl logs m1-3bfe12ac-postgres-stanza-create-gbtgb

time="2021-04-13T20:01:19Z" level=info msg="pgo-backrest starts"
time="2021-04-13T20:01:19Z" level=info msg="debug flag set to false"
time="2021-04-13T20:01:19Z" level=info msg="backrest stanza-create command requested"
time="2021-04-13T20:01:19Z" level=info msg="s3 flag enabled for backrest command"
time="2021-04-13T20:01:19Z" level=info msg="command to execute is [pgbackrest stanza-create --db-host=192.1.2.3 --db-path=/pgdata/m1-3bfe12ac-postgres --repol-type=s3]"
time="2021-04-13T20:01:19Z" level=info msg="command is pgbackrest stanza-create --db-host=192.1.2.3 --db-path=/pgdata/m1-3bfe12ac-postgres --repol-type=s3 "
time="2021-04-13T20:05:21Z" level=error msg="command terminated with exit code 49"
time="2021-04-13T20:05:21Z" level=info msg="output=[]"
```

- Job status is set to `type: Failed` and `status: "True"`, with reason: `BackoffLimitExceeded`:

```
kubectl get job m1-3bfe12ac-postgres-stanza-create -o yaml

status:
  conditions:
  - lastProbeTime: "2021-04-13T20:28:03Z"
    lastTransitionTime: "2021-04-13T20:28:03Z"
    message: Job has reached the specified backoff limit
    reason: BackoffLimitExceeded
    status: "True"
    type: Failed
    failed: 7
  startTime: "2021-04-13T20:01:18Z"
```

- If pod logs are lost, you can execute a command inside the `backrest-shared-repo` pod:
 - Obtain the `backrest-shared-repo` pod name:

```
kubectl get pods -n <namespace> | grep backrest-shared-repo
```

For example:

```
kubectl get pods | grep backrest-shared-repo
m1-eb8edc18-postgres-backrest-shared-repo-98dd46cc6-tw195 1/1 Running
```

- Exec into the pod:

```
kubectl exec -it <backrest-shared-repo-pod> -- bash
```

- Run:

```
pgbackrest info --output json --repol-type s3
```

For example, for invalid hostname:

```
ERROR: [049]: unable to get address for 'test1-v10-backup.s3.exampleaws.com': [-2] Name or service not knownCopy
```

The exact **ERROR**: will vary depending on the misconfiguration. Common errors:

- Invalid access key
- Invalid access key secret
- Invalid bucket region
- Invalid bucket or folder path

- To fix the incorrect settings, see [Reconfiguring or adding backup settings after installation of the management subsystem](#).

Restore failure with bootstrap pod error

Postgres bootstrap pod reports an error and restore does not progress.

Symptoms

The postgres bootstrap pod goes into an error state. Subsequent bootstrap pods come up showing **running**, but the restore does not progress. The logs of the postgres bootstrap pod show:

```
2022-08-01 15:03:58,337 INFO: Lock owner: None; I am management-09384302-postgres-bootstrap-v6xwz
2022-08-01 15:03:58,337 INFO: not healthy enough for leader race
2022-08-01 15:03:58,337 INFO: bootstrap in progress
ERROR: [125]: remote-0 process on 'management-09384302-postgres-backrest-shared-repo.e2e-automation.svc.cluster.local.' terminated unexpectedly [255]: ssh: connect to host management-09384302-postgres-backrest-shared-repo.e2e-automation.svc.cluster.local. port 2222: Connection timed out
Mon Aug 1 15:03:58 UTC 2022 ERROR: pgBackRest primary Creation: pgBackRest restore failed when creating primary
2022-08-01 15:03:59,010 INFO: removing initialize key after failed attempt to bootstrap the cluster
2022-08-01 15:03:59,026 INFO: renaming data directory to /pgdata/management-09384302-postgres_2022-08-01-15-03-59
Traceback (most recent call last):
  File "/usr/local/bin/patroni", line 11, in <module>
    sys.exit(main())
```



```

File "/usr/local/lib/python3.6/site-packages/patroni/__init__.py", line 171, in main
    return patroni_main()
File "/usr/local/lib/python3.6/site-packages/patroni/__init__.py", line 139, in patroni_main
    abstract_main(Patroni, schema)
File "/usr/local/lib/python3.6/site-packages/patroni/daemon.py", line 100, in abstract_main
    controller.run()
File "/usr/local/lib/python3.6/site-packages/patroni/__init__.py", line 109, in run
    super(Patroni, self).run()
File "/usr/local/lib/python3.6/site-packages/patroni/daemon.py", line 59, in run
    self._run_cycle()
File "/usr/local/lib/python3.6/site-packages/patroni/__init__.py", line 112, in _run_cycle
    logger.info(self.ha.run_cycle())
File "/usr/local/lib/python3.6/site-packages/patroni/ha.py", line 1469, in run_cycle
    info = self._run_cycle()
File "/usr/local/lib/python3.6/site-packages/patroni/ha.py", line 1343, in _run_cycle
    return self.post_bootstrap()
File "/usr/local/lib/python3.6/site-packages/patroni/ha.py", line 1236, in post_bootstrap
    self.cancel_initialization()
File "/usr/local/lib/python3.6/site-packages/patroni/ha.py", line 1229, in cancel_initialization
    raise PatroniFatalException('Failed to bootstrap cluster')
patroni.exceptions.PatroniFatalException: 'Failed to bootstrap cluster'

```

Resolution

Delete the Kubernetes API Connect ManagementCluster oplock and restore job, then start the restore again:

```

kubect1 delete cm <cr name>-oplock
kubect1 delete mgmtr <restoreCR name>

```

Backing up and restoring the Developer Portal in a Kubernetes environment

You can back up and restore your Developer Portal service in your Kubernetes environment.

Before you begin

Important: If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. The backups of the Management and Portal must be taken at the same time to ensure that the Portal sites are consistent with Management database.

About this task

To complete a backup and restore of your Developer Portal, you must have the **custom resource** configured with the **portalBackup** subsection. Backups can be scheduled or generated manually (on-demand). You can then list what backups are available and then use one to restore your Developer Portal if required. Ensure that your backup **path** has the correct permissions to allow your backup user to be able to read and write to it from within the Developer Portal. CR, **custom resource**, templates are provided in the **helper_files** archive to install each API Connect subsystem. You decompressed these files into an installation folder of your choosing in [Deploying operators and cert-manager](#).

Note: For points to consider when you are backing up and restoring a two data center disaster recovery deployment on Kubernetes, see [Backup and restore requirements for a two data center deployment](#).

The default Developer Portal backup schedule is once every 24 hours, but the schedule can be changed in the backup settings. The Developer Portal saves all system and site backups locally, and also saves them remotely based on the configured SFTP and s3 settings.

The local backups are automatically maintained so that the latest three backups of each site and of the system are kept, and older backups are removed. This maintenance means that the Developer Portal retains the latest three backups for each site and for the system however old they are, but there is no deletion of the old backups on the remote server. If a site is deleted, then all of the local backups for that site are also deleted, as otherwise the backup volume might become full of old site backups. For remote backups, you can configure a retention policy on your remote server to remove the old backup files as required.

Tip: If you are using remote backups, and there are a large number of rows in your **portalbackup** list that have a **CR TYPE** of **record**, there might be a slow down in system performance, including during Developer Portal upgrades. Although there is no upper limit to how many portal backups you can have, it is recommended that you prune your remote backups so that the number of rows does not become excessively large. One **portalbackup** row of type **record**, corresponds to one backup file on the configured remote backup server. For more information, see [Overview of the Developer Portal backup resources](#).

Procedure

- If you haven't already, configure your Developer Portal **custom resource** with the **portalBackup** subsection.

- Create your Developer Portal backup secret.

The backup secret is a Kubernetes secret that contains your credentials for accessing the s3 or SFTP backup database.

- s3

Only password-based authentication is supported for s3, not authentication based on public certificates and private keys. Password-based authentication for s3 requires that you generate an access key and secret. For example:

- IBM (Cloud Object Storage): [Service credentials](#).
- AWS: [Managing access keys](#).

The secret can be created with the following command:

```

kubect1 create secret generic portal-backup-secret --from-literal=username='<YOUR ACCESS KEY OR USER NAME>'
--from-literal=password='<YOUR ACCESS KEY SECRET OR PASSWORD>' -n <namespace-of-portal-subsystem>

```

- SFTP

Supported credentials types:

- Username and password (v10.0.2.0 or later).

- Username and SSH-key (v10.0.3.0 or later). Only OpenSSH keys are supported.¹
- Use one of the following commands to create the secret:

- Username and password credentials

```
$ kubectl create secret generic portal-backup-secret --from-literal=username='<YOUR USERNAME>'
--from-literal=password='<YOUR PASSWORD>' -n <namespace-of-portal-subsystem>
```

- Version 10.0.3.0 or later: Username and SSH-key credentials:

```
$ kubectl create secret generic portal-backup-secret --from-literal=username='<YOUR USERNAME>'
--from-file=ssh-privatekey='<YOUR PRIVATEKEY FILE>' -n <namespace-of-portal-subsystem>
```

2. Ensure that your Developer Portal `custom` resource is configured with the `portalBackup` subsection.

- All of your settings must be added in the `spec`: section of the file.
- To schedule automated backups, specify values for the `schedule` setting.

An example when you use `protocol: objstore`.

```
spec:
  portalBackup:
    credentials: portal-backup-secret
    host: s3.eu-gb.cloud-object-storage.appdomain.cloud/
    path: test-bucket/restore-test
    port: 443
    protocol: objstore
    backupCerts: <custom-s3-server-CA-cert>
    backups3URISyle: <host-or-path>
    schedule: '0 2 * * *'
```

An example when you use `protocol: sftp`.

```
sftp.
spec:
  portalBackup:
    credentials: portal-backup-secret
    host: sftp-service.example.com
    path: /home/fvtuser/site-backups
    port: 22
    protocol: sftp
    schedule: '0 2 * * *'
```

Notes about `portalBackup`: entry:

- If `portalBackup`: entry is not present or contain nothing, just `local` backups will be taken if a backup CR is created. If the `schedule` exists, backups are taken at the scheduled time.
- For S3 and SFTP, `portalBackup`: entry must contain host, credentials, protocol, and path.
- For s3, the `portalBackup`: section can optionally also contain either, or both, of `backupCerts` and `backups3URISyle`.

Table 1. Developer Portal backup settings

Setting	Description
<code>credentials</code>	The name of the secret you created. Supported credential types: <ul style="list-style-type: none"> • s3 Only password-based authentication is supported for s3, not authentication based on public certificates and private keys. Password-based authentication for s3 requires that you generate an access key and secret. • SFTP <ul style="list-style-type: none"> • Username and password (v10.0.2.0 or later) • Username and SSH-key (v10.0.3.0 or later) To create a secret, see step 1.
<code>host</code>	<ul style="list-style-type: none"> • s3 - <code>host</code>: is the S3 endpoint with the corresponding S3 region in the format <code><S3endpoint></code>, or with an optional S3 region in the format <code><S3endpoint>/<S3region></code>. • sftp - the hostname of the sftp server
<code>path</code>	<ul style="list-style-type: none"> • S3 - the name of your S3 bucket to store backup data, optionally followed by a / and then any subdirectories inside the bucket. Valid examples: <pre>bucket bucket/path bucket/lots/of/paths</pre> • sftp - the full absolute path of the folder on the SFTP server, beginning with /.
<code>port</code>	The port for the protocol to connect to the host. <ul style="list-style-type: none"> • S3 - Defaults to 443 if not specified. • SFTP - Defaults to 22 if not specified. • local - Ignored for local.
<code>protocol</code>	The protocol that is used to communicate with your remote backup endpoint. Specify one of the following values: <ul style="list-style-type: none"> • <code>sftp</code> - for secure file transfer protocol. • <code>objstore</code> - for S3 compatible object storage.

Setting	Description
backupCerts	<p>(S3 storage only) backupCerts is the name of a custom certificate that contains your upstream custom S3 CA certificate. Supported beginning with 10.0.2.</p> <p>Note: If you use a public certificate for the S3 storage provider, it must be signed by a known certificate authority that is trusted by API Connect. Use of an untrusted authority can cause the following error during backup upload:</p> <pre>x509: certificate signed by unknown authority</pre> <p>This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be ca.crt and the value is the base64-encoded value of the CA certificate.</p> <p>If the server certificate that is presented by your S3/object-store endpoint is signed by a well known CA and includes any intermediate certificates in the chain, then you do not need to provide the CA certificate using this feature because the Portal subsystem already trusts the server certificate. You can test this by configuring the Portal backup to your S3/object-store server without providing the CA certificate.</p> <p>If you do need to provide the CA certificate then you must provide the entire chain in the ca.crt member of the secret, as follows:</p> <pre>intermediate-certificate-1 intermediate-certificate-2 . . intermediate-certificate-n root-certificate</pre> <p>Where the first certificate in the ca.crt member (intermediate-certificate-1) is the issuer of the S3/object-store server certificate, and each subsequent certificate in the ca.crt member is the issuer of the preceding certificate. The root certificate in the ca.crt member must be self-signed.</p> <p>If there are no intermediate certificates involved, then the ca.crt member contains only the root certificate.</p> <p>The following sample bash script creates the Kubernetes secret:</p> <pre>cat >customs3ca.yaml <<EOF apiVersion: v1 data: ca.crt: \$(base64 <path-to-ca-certificate> tr -d '\n') kind: Secret metadata: name: custom-server-ca type: generic EOF</pre> <pre>kubectl apply -f customs3ca.yaml -n <namespace></pre>
backups3URIS style	<p>(S3 storage only) Optional - backups3URISstyle indicates whether URIs to your S3 backup should use specify a host or a path value. When not specified, default to host. Supported beginning with 10.0.2.</p> <p>Valid values: host and path.</p> <p>Some custom S3 providers require URI style to be set to path. For example, MinIO supports both host and path style setup. You can create a MinIO S3 server to only accept path style client communications.</p> <p>Important: Contact your custom S3 administrator before configuring this field. If not properly configured, an upstream custom S3 can reject connections from the client, such as the API Connect Management subsystem.</p>
schedule	<p>The schedule for how often automatic Developer Portal backups are run. The format for the schedule is any valid cron string. The timezone for backups is UTC.</p> <p>Valid for all backup types: s3, sftp, and local.</p> <pre>spec: portalBackup: . . schedule: '0 2 * * *'</pre>
portalBackup RecordDays	<p>The portalBackupRecordDays setting denotes the number of days that backup records (backups that exist on the remote backup server) are shown when you list your remote backups. It is defaulted to 30 days.</p>

- Reconfiguring Developer Portal backups

You can edit the deployed Developer Portal **custom resource**. You might want to do an edit if you did not include a **portalBackup** section before the installation. Or, if you made a mistake in your **portalBackup**, or if you'd like to change the **portalBackup** config. To edit your **portalcluster** CR:

1. Get the name of your **portalcluster** CR:

```
kubectl get portalcluster -n <namespace-of-portal-subsystem>
```

2. Edit your **portalcluster** CR:

```
kubectl edit portalcluster <name-of-portal-cr> -n <namespace-of-portal-subsystem>
```

3. Inside the **spec**, add the **portalBackup** section as explained in step 2.
4. Write and quit **wq**.

- How to generate a manual backup.

You can use these steps to configure a Developer Portal with backups that use IBM's Cloud Object Storage S3, AWS S3, or SFTP.

Decide which of the three types of manual backups you want to do:

- o **all**: backs up the Developer Portal system and all its sites to the remote server
- o **system**: backs up the Developer Portal system to the remote server
- o **site**: backs up a Developer Portal site to the remote server

1. You generate a backup manually by creating a **portalBackup custom resource** that is detected by the **ibm-apiconnect** operator that then triggers the backup process.

You can use **portalbackup_cr.yaml** in the **helper_files** directory as an example:

```

apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalBackup
metadata:
  generateName: portal-bup-
spec:
  type: all
  portalCluster: portal
  crType: create
  comment: "test comment"

```

- **type**: is the type of backup to be done. Valid arguments are:
 - all
 - system
 - site
- **portalCluster**: The name of the portal cluster you want to backup. You can find this with:


```
kubectl get portalcluster -n <namespace>
```
- **crType**: Use **create** to trigger a backup process. When the backup is created, the operator generates another backup CR with **crType**: **record**, which represents the stored backup that you can use for a restore operation.
- **comment**: you can optionally add a comment to the CR before you create a backup.
- **siteName**: used only when you take a site backup. Valid arguments are:
 - site uuid or url - back up that site only to the remote server
 - **installed** - backups all installed sites to the remote server

2. Create the **PortalBackup** custom resource in the **namespace** of your Developer Portal to trigger the backup process:

```
kubectl create -f portalbackup_cr.yaml -n <ns-of-portal-subsystem>
```

3. Optional: You can list current backups by using the command:

```
kubectl get portalbackup -n <ns-of-portal-subsystem>
```

Here is an example of some possible output:

NAME	ID	STATUS	TYPE	CR TYPE	AGE	COMMENT
p1-bup-gqz8f	20200526.142449	Ready	system	record	55m	
p1-bup-jx4vc	20200526.142458	Ready	site	record	54m	
p1-bup-st4l2		Ready	all	create	56m	y
p1-bup-th2dm		Ready	all	create	4h51m	y

Tip: If you want to order your **portalbackups** by ID, you can use the Kubernetes keyword **--sort-by**. For example:

```
kubectl get pb -n <ns> --sort-by=.status.backupId
```

4. Optional: You can get a detailed view of a specific backup by using the command:

```
kubectl get portalbackup <bup-name> -n <ns-of-portal-subsystem> -o yaml
```

Here is an example of some possible output:

```

apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalBackup
metadata:
  creationTimestamp: "2020-10-01T10:00:16Z"
  generateName: portal-bup-
  generation: 1
  name: portal-bup-g66gv
  namespace: default
  resourceVersion: "8940"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/default/portalbackups/portal-bup-g66gv
  uid: b1d5b295-b966-48f2-b6ef-dfcd1c499741
spec:
  comment: test comment
  portalCluster: portal
  crType: create
  siteName: installed
  type: site
status:
  backupId: ""
  commentLeft: "y"
  conditions:
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    message: |-
      2020-10-01 10:00:18: CLI task (backup:site) starting.
      2020-10-01 10:00:20: Making a backup for site conor.com
      2020-10-01 10:00:27: A local backup was successfully created: /var/aegir/backups/conor.com-20201001.100021.tar.gz
      2020-10-01 10:00:30: Attempting to upload backup to remote backup server
      2020-10-01 10:00:30: Uploading conor.com-20201001.100021.tar.gz to s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard:22:cq-test/ptlcq4/ using objstore
      2020-10-01 10:00:33: Site with URL conor.com (conor.com-20201001.100021.tar.gz) successfully uploaded
      2020-10-01 10:00:33: All installed sites have been successfully backed up
      2020-10-01 10:00:33: CLI task (backup:site) completed successfully.
    reason: BackupComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    reason: BackupInProgress

```

```

    status: "False"
    type: Running
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupStatusUnknown
    status: "False"
    type: Warning
  fullBackupName: ""
  id: f8t0ux7z4zfnjnju
  message: ""
  phase: Ready
  state: ""

```

- How to list available Developer Portal backups on remote server.
Use command:

```
kubectl get portalbackups -n <ns-of-portal-subsystem>
```

Here is an example of some possible output:

```
kubectl get portalbackup -n <ns-of-portal-subsystem>
```

NAME	ID	STATUS	TYPE	CR TYPE	AGE	COMMENT
p1-bup-gqz8f	20200526.142449	Ready	system	record	55m	
p1-bup-jx4vc	20200526.142458	Ready	site	record	54m	
p1-bup-st4l2		Ready	all	create	56m	y
p1-bup-th2dm		Ready	all	create	4h51m	y

Note: You can only restore the Portal subsystem from backups where the **CR TYPE** is **record**.
Optionally, if you need more information, such as the name of the .tgz file that you can run, use command:

```
kubectl get portalbackups <bup-name> -n <ns-of-portal-subsystem> -o yaml
```

Here is an example of some possible output:

```

apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalBackup
metadata:
  creationTimestamp: "2020-10-01T10:01:19Z"
  generateName: portal-bup-
  generation: 1
  name: portal-bup-z7wgk
  namespace: default
  resourceVersion: "8938"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/default/portalbackups/portal-bup-z7wgk
  uid: 5c5899c0-4da0-4468-b7fb-1c283121a053
spec:
  comment: ""
  portalCluster: portal
  crType: record
  siteName: ""
  type: site
status:
  backupId: "20201001.100021"
  commentLeft: ""
  conditions:
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    reason: BackupFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    message: Record Backup CR for conor.com-20201001.100021.tar.gz
    reason: BackupComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    reason: BackupInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    reason: BackupStatusUnknown
    status: "False"
    type: Warning
  fullBackupName: conor.com-20201001.100021.tar.gz
  id: ""
  message: ""
  phase: Ready
  state: ""

```

- How to restore a Developer Portal subsystem.
Ensure that the backup method that you used is complete before you attempt a restore. The backup data that is stored in the remote server is used for restoring the Developer Portal subsystem data back to a previous state. Ensure that these backups are on your remote server before you attempt a restore.

There are three types of restores:

- **all**: restores the Developer Portal system and all the sites that are found on the remote server by using the latest backup.
- **system**: restores the Developer Portal system to the remote server
- **site**: restores the Developer Portal site from the remote server

1. Decide on the options to use for your **PortalRestore custom resource**. The **PortalRestore custom resource** is detected by the **ibm-apiconnectoperator** that then triggers the backup process.

You can use `portalrestore_cr.yaml` in the `helper_files` directory as an example:

```

apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalRestore
metadata:
  generateName: portal-restore-
spec:

```

```

type: all
portalCluster: portal
dryRun: true
timestamp: now
priorityList:
- myportal-critical-site.com
- myportal-important.com

```

- **type**: is the type of restore to be done. Valid arguments are:
 - all
 - system
 - site
- **portalCluster** is the name of the portal cluster you want to restore. You can find this with:


```
kubectl get portalcluster -n <namespace>
```
- **siteName**: used only when you took a site backup. Valid arguments are:
 - **ID**, for example 20200526.142458: the backup ID of a portal backup CR. The ID must be taken from a portal backup that is of **type site**, and **crType record**.
 - **site-1234.tgz**: restore the site by using this file name that exists on the local file system or remote server.
 - **myportal.com**: find the latest backup for this URL (locally or remotely) and restore it.
 - **all**: restore all sites that are found on the remote server by using the latest backup.
 - **site** uuid or url - back up that site only to the remote server
 - **installed** - backups all installed sites to the remote server.
- **systemName** - Valid arguments are:
 - **ID**, for example 20200526.142449: the backup ID of a portal backup CR. The ID must be taken from a portal backup that is of **type system**, and **crType record**.
 - **system_backup-1234.tgz**: restores the system from the given file that exists on the pod's local file system or the remote server.
 - **latest**: restores from latest system backup that is found from either local file system or remote server.
- **timestamp** - Valid arguments are:
 - **now**: use the latest backups available.
 - **< TIMESTAMP >**: in 'YYYYMMDD.HHMMSS' format, specify a timestamp to retrieve the backup from. The nearest backup, searching backwards from this timestamp, is used.

Note: From IBM® API Connect Version 10.0.3.0, the timestamp format changed to **YYYYMMDD.HHMMSS**. For Version 10.0.2.0 and earlier, the timestamp format is **YYYY-MM-DD HH:MM:SS**.
- **dryRun** - Valid arguments are:
 - **true**: runs a dry run of the command, returning the actions that will be taken. These actions can be found in the custom resource by following step 4.
 - **false**: runs the command, restoring and replacing the system files and all sites. Any existing sites are reinstalled.
- **priorityList** - Prioritizes the restore of any sites. The sites are listed in the order in which they are provided. They are queued for restore first, followed by any remaining sites present in the backup location. Valid arguments are:
 - **myportalsite.com**
 - **mysecondportalsite.com**
- **customPlatformApiHostname** - If the API hostname of your management subsystem platform has changed since taking the backup you are restoring, then update it to the new platform API hostname of the management subsystem. (Optional)

Table 2. The options that are needed for each Developer Portal restore

	all	system	site
type	X	X	X
portalCluster	X	X	X
siteName			X
systemName		X	
timestamp	X		
dryRun	X		
priorityList	X		

2. Create the **PortalRestore** custom resource in the **namespace** of the Developer Portal subsystem to trigger the restore process:

```
kubectl create -f portalrestore_cr.yaml -n <ns-of-portal-susbystem>
```

3. Optional: You can list current restores by using the command:

```
kubectl get portalrestore -n <ns-of-portal-susbystem>
```

Here is an example of some possible output:

```

NAME                READY   AGE
portal-restore-twjrp Ready   28s

```

4. Optional: You can get a detailed view of a specific restore by using the command:

```
kubectl get portalrestore <restore-name> -n <ns-of-portal-susbystem> -o yaml
```

Here is an example of some possible output:

```

apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalRestore
metadata:
  creationTimestamp: "2020-11-11T10:14:39Z"
  generateName: portal-restore-
  generation: 1
  name: portal-restore-8f4c5
  namespace: portal

```

```

resourceVersion: "1497597"
selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/portal/portalrestores/portal-restore-8f4c5
uid: 5a70b657-d998-4989-b9b6-8480e986311d
spec:
  siteName: "20200526.090209"
  portalCluster: portal
  type: site
status:
  backupId: ""
  commentLeft: ""
  conditions:
  - lastTransitionTime: "2020-11-11T10:14:39Z"
    reason: RestoreFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-11-11T10:14:39Z"
    reason: RestorePending
    status: "False"
    type: Pending
  - lastTransitionTime: "2020-11-11T14:55:32Z"
    message: |-
      2020-05-26 13:11:25: CLI task (restore:site) starting.
      2020-05-26 13:11:36: Deleting any old site data if it exists...
      2020-05-26 13:13:18: Restoring site test.com ...
      2020-05-26 13:13:18: .
      2020-05-26 13:13:19: Patching the database dump before restoring ...
      2020-05-26 13:13:48: .
      2020-05-26 13:19:41: Triggering the drush command to restore the site. This may take some time.
      2020-05-26 13:19:48: .
      2020-05-26 13:25:40: Drush restore command completed. Setting up the site.
      2020-05-26 13:25:48: .
      2020-05-26 13:30:49: Site test.com restored from backup /var/aegir/backups/test.com-20200526.090209.tar.gz.
    Initializing the site...
      2020-05-26 13:30:52: Restore site completed.
      2020-05-26 13:30:52: CLI task (restore:site) completed successfully.
    reason: RestoreComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-11-11T14:55:32Z"
    reason: RestoreInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-11-11T10:14:39Z"
    reason: RestoreStatusUnknown
    status: "False"
    type: Warning
  fullBackupName: ""
  id: bybyibd99wxtmkmk
  message: ""

```

- Troubleshooting

1. Check the Developer Portal backup CR or restore CR.

Use command:

```
kubectl get <portalbackup/portalrestore> <bup-name/restore-name> -n <namespace> -o yaml
```

Here is an example of some possible output:

```

apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalRestore
metadata:
  creationTimestamp: "2020-05-26T13:10:46Z"
  generateName: portal-restore-
  generation: 1
  name: portal-restore-x2bf9
  namespace: portal
  resourceVersion: "1493471"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/portal/portalrestores/portal-restore-x2bf9
  uid: 9f6ale22-55ab-4595-8b98-bdfaec40d58c
spec:
  portalCluster: portal
  type: site
status:
  backupId: ""
  commentLeft: ""
  conditions:
  - lastTransitionTime: "2020-05-26T13:10:49Z"
    message: |
      Please provide a site to restore. To restore all sites use value 'all' in the CR
    status: Failed
    type: Ready
  fullBackupName: ""
  id: ""
  message: ""

```

Read the message inside the conditions of the Developer Portal backup CR or the restore CR.

2. Retrieving logs

Look for information in the logs of the `ibm-apiconnect` pod:

- Locate the `ibm-apiconnect` pod with:

```
kubectl get pods -n <namespace-of-portal-subsystem>
```

- View the logs with:

```
kubectl logs <ibm-apiconnect-pod-NAME> -n <namespace-of-portal-subsystem>
```

- [Overview of the Developer Portal backup resources](#)
How to manage your Developer Portal remote backup resources.

¹ PuTTY style keys can be converted to OpenSSH by using the PuTTY Key Generator (PuTTYgen) application; see <https://www.puttygen.com/>.

Overview of the Developer Portal backup resources

How to manage your Developer Portal remote backup resources.

Note: This information is relevant only if you're taking remote Developer Portal backups.

The Developer Portal uses two types of portal backup resources. When you run the command `kubectl get portalbackup`, you see a list of backups, for example:

NAME	ID	TASK ID	STATUS	TYPE	CR TYPE	CLUSTER	AGE	COMMENT
my-backup-f91ms	n/a	20b47fe090f5	Ready	system	create	portal	78m	
portal-record-222cg	20220108.010033		Ready	system	record	portal	94m	
portal-record-22xdc	20220702.010339		Ready	site	record	portal	9m49s	
portal-record-245j7	20221002.010340		Ready	site	record	portal	5m	
portal-record-246hm	20221218.010103		Ready	site	record	portal	32m	
portal-record-24h44	20220509.010037		Ready	site	record	portal	68m	
portal-record-24wb4	20221003.010029		Ready	site	record	portal	60m	
portal-record-264mh	20221119.010453		Ready	system	record	portal	78m	

There are two types of resources listed, and these are denoted by the **CR TYPE** column; `create` or `record` types.

1. **create**: This resource is a representation of a `portalbackup` task that was created manually. See [Backing up and restoring the Developer Portal in a Kubernetes environment](#), [Backing up Developer Portal on OpenShift and Cloud Pak for Integration](#), or [Backing up the Developer Portal subsystem](#) on VMware, for information about how to create a manual portal backup.
2. **record**: This resource is generated by the operator, and it's a representation of a backup file that's stored on the remote backup server. A record listing can be modified only to add a comment to it. You can describe each record to understand which file it is referring to by running the following command:

```
kubectl describe portalbackup portal-record-name
```

For example:

```
kubectl describe portalbackup portal-record-zmmfw
.
.
Name:          portal-record-zmmfw
Namespace:    apic
Labels:       createdBy
.
Full Backup Name:  my-portal-s3-hostname@my-porg@catal-20220604.010302.tar.gz
```

Record rows are generated and updated on a schedule, according to the backups that exist on your remote backup server. If you have 10 backup files stored on your remote server, then there will be 10 record rows; one row for each file.

The Developer Portal doesn't have a retention policy for remote backups. Therefore, it's your responsibility to ensure that you're keeping the number of backups that are stored remotely under control. An excessive number of portal backups results in delays administrating and upgrading the Developer Portal. This is due to the operator needing to maintain the excessive number of record rows that correspond to each remote portal backup.

Backing up and restoring the analytics database

The IBM® API Connect Analytics database can be backed up and restored from an S3 repository. S3 compatible object storage is required; for example, IBM Cloud Object Storage.

About this task

Before you can initiate a backup or restore operation, you must update the Analytics CR (custom resource) to configure your settings. Generating a backup for Analytics is performed on demand by deploying a backup CR. You can deploy a restore CR to start the restoration of a previously generated backup.

Note: On a v10.0.5 API Connect deployment, you can restore only analytics backups that were taken from a v10.0.5 analytics deployment. You cannot restore analytics backups that were taken from a deployment that was on v10.0.4 or an earlier release.

- [Configuring backup settings for Analytics](#)
Configure the IBM API Connect Analytics subsystem on Kubernetes to support backing up and restoring the Analytics database.
- [Running a backup of the Analytics database](#)
Deploy the `analyticsbackup_cr.yaml` CR to initiate a backup of the IBM API Connect Analytics database on Kubernetes.
- [Restoring the Analytics database](#)
Deploy the `analyticsrestore_cr.yaml` CR to initiate restoration of a backup of the IBM API Connect Analytics database on Kubernetes.
- [Troubleshooting Analytics backup and restore](#)
If you encounter problems when backing up or restoring the IBM API Connect Analytics database on Kubernetes, review the following troubleshooting tips.

Configuring backup settings for Analytics

Configure the IBM® API Connect Analytics subsystem on Kubernetes to support backing up and restoring the Analytics database.

About this task

The Analytics database can be backed up and restored from an S3 repository. S3-compatible object storage is required; for example, IBM Cloud Object Storage. Generating a backup for Analytics is performed on demand by creating a backup CR (custom resource). You can create a restore CR to start the restoration of a previously generated backup.

Note:

- Only host-style S3 backups are supported. Path-style S3 backups are not supported.
- If you are using a self-managed S3 backup server, check that you have a DNS hostname entry and matching TLS certificate for your S3 backup endpoint.

Note: You can only run one backup or restore process at a time.

Procedure

1. Create the analytics backup secret for your S3-compatible storage:

a. Obtain an access key and corresponding access key secret from your S3 provider.

For example, you might obtain the key and secret from one of the following providers, using the instructions found on their sites:

- [IBM Cloud Object Storage](#)
- [AWS](#)

b. Create the Kubernetes secret by running the following command and filling in your access key, access key secret, and namespace:

```
kubectl create secret generic analytics-backup-secret --from-literal=access_key='Your_Access_Key' --from-literal=secret_key='Your_access_key_secret' -n Namespace_of_Analytics_Subsystem
```

2. Add the backup configuration to the Analytics CR:

a. Open the analytics_cr.yaml file for editing.

If you already installed Analytics without configuring backup settings, run the following command to open the deployed version of the Analytics CR:

```
kubectl edit analyticscluster analytics_cr -n namespace
```

where:

- `analytics_cr` is the name of the YAML file containing the Analytics CR
- `namespace` is the name of the Analytics namespace

b. In the `spec` section, add a `databaseBackup` section and configure the backup settings as shown in the following example:

```
spec:
  databaseBackup:
    credentials: analytics-backup-secret
    host: s3.eu-gb.cloud-object-storage.appdomain.cloud
    path: my-s3-bucket/apic-analytics-backups
    backupCerts: <custom-s3-server-CA-cert>
    schedule: "0 2 * * *"
    enableCompression: true
    chunkSize: 1GB
    enableServerSideEncryption: false
```

where:

- `credentials` is the name of the Kubernetes secret you created in step 1
- `host` is the S3 endpoint with the corresponding S3 region in the format:

```
s3.s3region.s3domain
```

Attention: A path style host value (formatted as `S3endpoint/S3region`) is not supported.

- `path` is a combination of the S3 bucket and the base path within the bucket, using the format: `bucket_name/path`
- `backupCerts` is the name of a custom certificate that contains your upstream custom S3 CA certificate. Supported beginning with 10.0.2. This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be `ca.crt` and the value is the base64-encoded value of the CA certificate.

If the server certificate that is presented by your S3/object-store endpoint is signed by a well known CA and includes any intermediate certificates in the chain, then you do not need to provide the CA certificate using this feature because the Analytics subsystem already trusts the server certificate. You can test this by configuring the Analytics backup to your S3/object-store server without providing the CA certificate.

If you do need to provide the CA certificate then you must provide the entire chain in the `ca.crt` member of the secret, as follows:

```
intermediate-certificate-1
intermediate-certificate-2
.
.
intermediate-certificate-n
root-certificate
```

Where the first certificate in the `ca.crt` member (intermediate-certificate-1) is the issuer of the S3/object-store server certificate, and each subsequent certificate in the `ca.crt` member is the issuer of the preceding certificate. The root certificate in the `ca.crt` member must be self-signed.

If there are no intermediate certificates involved, then the `ca.crt` member contains only the root certificate.

The following sample bash script creates the Kubernetes secret:

```
cat >customs3ca.yaml <<EOF
apiVersion: v1
data:
```

```

ca.crt: $(base64 <path-to-ca-certificate> | tr -d '\n' )
kind: Secret
metadata:
  name: custom-server-ca
  type: generic
EOF

```

```
kubect1 apply -f customs3ca.yaml -n <namespace>
```

- **schedule** is the schedule that determines how often backups will be invoked automatically. The format for the schedule is any valid **cron** string. The timezone for backups is that of the kube-controller-manager. Backing up once a day is probably sufficient; remember that the more frequently you run backups, the more storage space you need.
 - Optional: **enableCompression** determines whether metadata files are stored in compressed format. Default value is true.
 - Optional: **chunkSize** specifies how large files are stored. Large files can be stored as chunks when the snapshot is created. This setting specifies the size of the chunks as GB, MB, or KB. Default value is 1GB.
 - Optional: **enableServerSideEncryption** determines whether files are encrypted. When set to true, files are encrypted on the server side using AES256. Default value is false.
3. Save and close the file.
 4. Install or modify the Analytics subsystem with the updated CR as explained in [Installing analytics](#).

Running a backup of the Analytics database

Deploy the analyticsbackup_cr.yaml CR to initiate a backup of the IBM® API Connect Analytics database on Kubernetes.

Before you begin

[Configure backup settings for Analytics](#).

About this task

A backup is triggered by creating the analyticsbackup_cr.yaml CR. The new CR is detected by the **ibm-apiconnect** operator, which requests a new backup.

Procedure

1. Create the **AnalyticsBackup** CR for your deployment in a file called analyticsbackup_cr.yaml.
For example:

```

apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsBackup
metadata:
  generateName: a7s-bup-
  namespace: <analytics namespace>
spec:
  crType: create
  comment: "test comment"
  indices:
  - all
  enableIgnoreUnavailable: true

```

where:

- **metadata.generateName** - Use to specify a prefix for your backup name, the remainder of the name will be generated at runtime.
 - **metadata.name** - Use as an alternative to **generateName**, where you want to specify the full name of the backup.
 - Optional: **metadata.namespace** - Specify the analytics namespace.
 - **spec.crType** - Use **create** to trigger a backup process. When the backup is created, the operator generates another backup CR with **crType: record**, which represents the stored backup that you can use for a restore operation.
 - Optional: **spec.comment** - You can use this field to describe the back up for later reference.
 - Optional: **spec.indices** - A list of index names or keywords, indicating which indices you want to back up. Options are: **all** | **apievents** | **ui**, or the names of individual indices. If no indices are specified, then all indices are backed up. The default of 'all' is specified explicitly in the above sample.
 - Optional: **spec.enableIgnoreUnavailable** - Default value is **false**. If set to **false**, the backup fails if a specified index is not present. If set to **true**, missing indices are skipped and the backup continues.
2. Deploy your CR by running the following command to create it in the namespace for your Analytics subsystem:

```
kubect1 create -f analyticsbackup_cr.yaml -n namespace
```

where **namespace** is the name of the Analytics namespace.

The backup's name and ID are generated when the backup operation runs.

3. To get a list of available backups, run the following command:

```
kubect1 get analyticsbackup -n namespace
```

The response looks like the following example:

NAME	STATUS	ID	CR TYPE	INDICES	AGE	COMMENT
a7s-bup-266p8	Ready		create	[all]	20h	y
a7s-bup-pffdh	Ready		create	[all]	34h	y

4. Display the details for a particular backup by running the following command:
Use the backup name from the results in step 4.

```
kubectl get analyticsbackup backup_NAME -n namespace -o yaml
```

The result looks like the following example:

```
apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsBackup
metadata:
  creationTimestamp: "2020-05-20T16:18:05Z"
  generateName: a7s-bup-
  generation: 1
  name: a7s-bup-gb6s5
  namespace: my-namespace
  resourceVersion: "867329"
  uid: f43b8400-733d-411a-9e9a-de2166bbfb0a
spec:
  indices:
  - all
status:
  conditions:
  - lastTransitionTime: "2022-06-30T16:18:24Z"
    message: SUCCESS
    status: Complete
    type: Ready
  details: |
    {Name:my-backup-id
    Version:5.6.16
    Indices:[...]
    StartTime:2022-06-30T16:18:09.698Z
    EndTime:2022-06-30T16:18:16.082Z
    State:SUCCESS
    Failures:[]
    Shards:{Failed:0
    Successful:1}}
  id: my-backup-id
```

Restoring the Analytics database

Deploy the analyticsrestore_cr.yaml CR to initiate restoration of a backup of the IBM® API Connect Analytics database on Kubernetes.

Before you begin

[Create a backup.](#)

About this task

A restore operation is triggered by creating the analyticsrestore_cr.yaml CR. The new CR is detected by the **ibm-apiconnect** operator, which requests a restore of the specified backup.

Note: You can only run one backup or restore process at a time.

Procedure

1. Get a list of available backups by running the following command:

```
kubectl get analyticsbackup -n namespace
```

where **namespace** is the name of the Analytics namespace.

The response looks like the following example. Note down the ID of the backup that you want to restore.

NAME	STATUS	ID	CR TYPE	INDICES	AGE	COMMENT
a7s-bup-266p8	Ready		create	[all]	20h	y
a7s-bup-pffdh	Ready		create	[all]	34h	y
analytics-bup-kxwq5	Ready	analytics-all-2020-09-07t03:49:15utc	record	[all]	34h	
analytics-bup-p29z8	Ready	analytics-all-2020-09-07t18:04:03utc	record	[all]	20h	

Note: You can only restore the Analytics subsystem from backups where the **CR TYPE** is **record**.

2. Create the **AnalyticsRestore** CR for your deployment in a file called analyticsrestore_cr.yaml.
The following code shows the sample analyticsrestore_cr.yaml, which is stored in the helper_files directory:

```
apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsRestore
metadata:
  generateName: a7s-restore-
spec:
  indices:
  - all
  backupID: analytics-all-2022-07-01t03:49:15utc
  enableIgnoreUnavailable: true
  enableOverride: false
```

3. In your copy of the CR, update the following settings:

- **indices** - A list of index names or keywords, indicating which indices you want to restore. Options are: **all** | **apievents** | **ui**, or the names of individual indices. If no indices are specified, then all indices are restored. The default of 'all' is specified explicitly in the above sample.
- **backupID** - The ID of the backup to be restored, which you obtained in step 2.
- **enableIgnoreUnavailable** - Default value is **false**. If set to **false**, the restore fails if a specified index is not present in the backup identified by the **backupID**. If set to **true**, missing indices are skipped and the restore continues.
- **enableOverride** - Default value is **false**. If set to **true**, then the restore overrides existing indices. When **enableOverride** is set to **false** and you specified indices that already exist in the database that you are restoring, then the restore fails. The failure prevents you from accidentally replacing or losing data that exists in the current database.

4. Deploy your CR by running the following command to create it in the namespace for your Analytics subsystem:

```
kubectl create -f analyticsrestore_cr.yaml -n namespace
```

where **namespace** is the name of the Analytics namespace.

The restore's name is generated when the restore operation runs.

5. To get a list of restores, run the following command:

```
kubectl get analyticsrestore -n namespace
```

The response looks like the following example.

NAME	STATUS	ID	AGE
a7s-restore-9z4fr	Ready	analytics-all-2022-07-01t04:49:15utc	1h

6. Display the details for a particular restore by running the following command:

Use the restore name from the results in step 6.

```
kubectl get analyticsrestore restore_NAME -n namespace -o yaml
```

The result looks like the following example:

```
apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsRestore
metadata:
  creationTimestamp: "2022-07-01T04:06:03Z"
  generateName: a7s-restore-
  generation: 1
  name: a7s-restore-9z4fr
  namespace: my-namespace
  resourceVersion: "975152"
  uid: fdda9105-17ab-44ac-83de-852620a4dcde
spec:
  backupID: analytics-all-2022-07-01t03:49:15utc
  enableIgnoreUnavailable: false
  enableOverride: true
  indices:
  - all
status:
  conditions:
  - lastTransitionTime: "2022-07-01T04:08:03Z"
    message: WARNING
    status: Warning
    type: Ready
  details: |
    &{NumberOfDataNodes:1
    Status:yellow
    ActivePrimaryShards:23
    ActiveShards:23
    RelocatingShards:0
    InitializingShards:0
    UnassignedShards:40
    PendingTasks:0}
  id: analytics-all-2022-07-01t04:49:15utc
```


When the restore is successfully started, the **status: condition** section displays **Running**. If the restore does not start correctly, the **status: condition** section displays **Failed**.

The restore process is successful when the storage cluster status is green, there are no unassigned shards, and there are no initializing shards. At that time the **status: condition** section displays **Complete**. If **status: Complete** is not achieved after 3 hours, the status changes to **Warning** and the details are no longer updated.

Important: For deployment profiles with fewer than 3 storage nodes, the restore process is never marked as **Complete**. The storage cluster status is expected to be yellow and to have more than 0 unassigned shards. In this situation, the restore process is likely to be complete when there are 0 initializing shards.

Troubleshooting Analytics backup and restore

If you encounter problems when backing up or restoring the IBM® API Connect Analytics database on Kubernetes, review the following troubleshooting tips.

Note: In the  Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.

Check the CR status

Look for messages in the **status** section of the backup or restore CR by running the following command:

```
kubectl get analytics[backup|restore] restore|backup_NAME -n namespace -o yaml
```

where:

- You specify whether to retrieve the **analyticsbackup** CR or the **analyticsrestore** CR
- **restore|backup_NAME** is the name that is generated when you initially deploy the backup or restore CR.
- **namespace** is the namespace where Analytics is deployed.

For example:

```
kubectl get analyticsbackup a7s-bup-gb6s5 -n analytics -o yaml
```

Check the logs

Look for information in the logs of the **ibm-apiconnect** pod:

1. Locate the **ibm-apiconnect** pod by running the following command:

```
kubectl get pods -n namespace
```

2. View the logs with:

```
kubectl logs ibm-apiconnect-pod-NAME -n namespace
```

If there is a problem with your **databaseBackup** configuration, there are messages with the label "error" to indicate the problem

Backups on OpenShift

You can backup and restore API Connect subsystems on OpenShift and Cloud Pak for Integration.

- [Backing up and restoring the management subsystem on OpenShift and Cloud Pak for Integration](#)
You can use the following instructions to backup and restore your Management subsystem on OpenShift and Cloud Pak for Integration.
- [Backing up and restoring Developer Portal on OpenShift and Cloud Pak for Integration](#)
You can back up and restore your Developer Portal service in your OpenShift or Cloud Pak for Integration environment.
- [Backing up and restoring the Analytics database on OpenShift and Cloud Pak for Integration](#)
The IBM® API Connect Analytics database can be backed up and restored from an S3 repository. S3 compatible object storage is required; for example, IBM Cloud Object Storage.

Backing up and restoring the management subsystem on OpenShift and Cloud Pak for Integration

You can use the following instructions to backup and restore your Management subsystem on OpenShift and Cloud Pak for Integration.

- To complete a backup and restore of your Management subsystem, you must have the **custom resource** configured with the **databaseBackup** subsection in the management subsystem custom resource.
- Backups can be scheduled or generated manually (on-demand). You can then list what backups are available and then use one to restore your Management subsystem if required.
- Postgres replica pods depend on a successful completed backup. If backup configuration is not correct, replica pods won't come up.
- In v10.0.1.1-eus and later, a new parameter called **protocol** is required in **databaseBackup** section
- Restoring a backup restores the registration credentials (**client_ID**, **client_secret**) that were in use at the time that the selected backup was created. For information on the registration credentials, see [Changing the registration client_id and client_secret for applications](#).

Note: You must backup both the Management and Portal subsystems at the same time, to ensure synchronicity across the services.

Types of backups supported

- Cloud-based S3 storage
 - IBM Cloud Object Storage
 - AWS S3
- SFTP
 - In v10.0.1.1-eus, SFTP backup/restore procedure is automatically handled by apiconnect operator.
 - In v10.0.1.0 and later, local Backups can be stored remotely on an sftp server, and retrieved later from the sftp server for restore operations. See [Using an sftp server for backup files \(v10.0.1.0\)](#).
- Local storage backups

Support for Cloud Pak for Integration

IBM Cloud Pak for Integration is deployed on OpenShift, and uses the API Connect distribution for OpenShift. The backup and restore instructions in this section apply to both API Connect on OpenShift, and API Connect within a Cloud Pak for Integration environment.

When you perform configuration tasks you can use either a graphical user interface or a command line. The IBM Cloud Pak Platform UI differs from the OpenShift web console. The configuration topics in this section provide instructions for both interfaces. Note that Cloud Pak for Integration uses the OpenShift command line utility `oc`. Command line instructions for configuration are identical for both platforms.

- [Configuring backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration](#)
You can configure backups for the API Connect Management subsystem that you will deploy in your OpenShift or Cloud Pak for Integration environment. Use the appropriate instructions for the type of backup that you want to configure:
- [Reconfiguring or adding backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration](#)
You can reconfigure the backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration.
- [Generating a manual backup of the management subsystem on OpenShift and Cloud Pak for Integration](#)
You can generate a manual (non-scheduled) backup of the management subsystem in your OpenShift and Cloud Pak for Integration environments.
- [Restoring the management subsystem on OpenShift and Cloud Pak for Integration](#)
You can restore your Management subsystem in your OpenShift and Cloud Pak for Integration environments.
- [Troubleshooting management subsystem backups on OpenShift and Cloud Pak for Integration](#)
You can troubleshoot failures of the API Connect management subsystem backups in an OpenShift or Cloud Pak for Integration environment.

Configuring backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration

You can configure backups for the API Connect Management subsystem that you will deploy in your OpenShift or Cloud Pak for Integration environment. Use the appropriate instructions for the type of backup that you want to configure:

- [Configuring S3 backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration](#)
Configure backups for the API Connect Management subsystem that you will deploy in your OpenShift or Cloud Pak for Integration environment.
- [Configuring SFTP backup settings for fresh install of the Management subsystem on OpenShift and Cloud Pak for Integration](#)
Beginning with API Connect V10.0.1.1, you can configure SFTP backups for your Management subsystem in your OpenShift and Cloud Pak for Integration environments.
- [Configuring local backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration](#)
Configure local backups for the API Connect Management subsystem that you will deploy in your OpenShift or Cloud Pak for Integration environment.

Configuring S3 backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration

Configure backups for the API Connect Management subsystem that you will deploy in your OpenShift or Cloud Pak for Integration environment.

Before you begin

If the `APIConnectCluster` instance is already created, do not use these instructions. Instead, add the Management backup configuration by following the instructions in [Reconfiguring or adding backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration](#).

About this task

When creating an instance of API Connect, use the advanced configuration settings to configure database backups for the Management subsystem. Beginning with v10.0.2, the Management subsystem supports custom S3 solutions such as MinIO.

Important: For S3 backups of the management subsystem, do not use retention features provided by Cloud-based S3 storage providers. Use of such features can result in periodic deletion of archived backups, which can cause backup corruptions which can cause database restores to fail.

Warning:

When configuring a new management subsystem for S3 backup, the content of the S3 bucket path must be empty. If you want to configure a management subsystem to use the content of an S3 bucket path already used by another management subsystem, you have two options:

1. Shutdown the postgres database on the existing management subsystem that points to the S3 bucket path before you start the new management subsystem.
2. In a disaster recovery scenario where you want to restore from the backup of an existing installation: Create a new S3 path and use S3 tools to copy the backup content from the existing management subsystem S3 bucket path, to the new bucket path.

Procedure

1. Create a backup secret.
The backup secret is a Kubernetes secret that contains your username and password for your S3 backup database.

For examples of how to generate the appropriate access key and secret for two of them, see:

- IBM (Cloud Object Storage): [Service credentials](#).
- AWS: [Managing access keys](#).

The secret can be created with the following command:

```
oc create secret generic mgmt-backup-secret --from-literal=key='<YOUR ACCESS KEY>' --from-literal=keysecret='<YOUR ACCESS KEY SECRET>' -n <namespace_of_management_subsystem>
```

2. Configure the Management backup settings.
Create the `APIConnectCluster` installation CR and add the Management backup configuration using one of the following methods:

- OpenShift web console:

- Navigate to Advanced Configuration > ManagementSubsystem > Advanced Configuration > Database Backups > Advanced Configuration
 - See the on-screen instructions for how to fill in each field. See also [Management subsystem settings](#).
Note that there is downtime involved when changing S3 protocol backup configuration.
 - IBM Cloud Pak Platform UI:
 - In the Platform UI for the API Connect cluster instance, select Configuration. On the Form UI tab, set Advanced Options to On.
 - See the on-screen instructions for how to fill in values for:
 - Credentials
 - Hostname
 - Path
 - Protocol
 - Restart DB (toggle on)
 - S3 Provider
 - Backup Credentials
 - Backup URI Style
 - Schedule (The timezone for backups is that of the node on which the postgres-operator pod is scheduled.)
 - Repo Retention
- For more information, see [Management subsystem settings](#).
- YAML file: Use Table 1 for guidance while you define the backup settings as shown in the following example.

```
spec:
  management:
    databaseBackup:
      protocol: objstore
      s3provider: ibm
      host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
      path: apic-backup
      credentials: mgmt-backup-secret
      schedule: "0 3 * * *"
      repoRetentionFull: 2
```

Note: The `protocol` setting was introduced in v10.0.1.1. Do not specify it on v10.0.1.0 or earlier deployments

Table 1. Backup configuration settings

Setting	Description
<code>protocol</code>	The type of the backup; required in 10.0.1.1-eus and greater. For s3 storage, use <code>objstore</code> . When the parameter is not set, the backup type defaults to local storage backup. Note: The <code>protocol</code> setting was introduced in v10.0.1.1. Do not specify it on v10.0.1.0 or earlier deployments
<code>s3Provider</code>	Name of the S3 provider to use. You must specify one of the supported values of <code>aws</code> , <code>ibm</code> , or <code>custom</code> . The <code>custom</code> option is supported beginning with 10.0.2 or later. If you use a custom S3 provider, you must also specify the <code>backupCerts</code> . Note: The public certificate on the S3 storage provider must be signed by a known certificate authority that is trusted by API Connect. Use of an untrusted authority can cause the following error during backup upload: <code>x509: certificate signed by unknown authority</code> .
<code>backupCerts</code>	The name of a custom certificate that contains your upstream custom S3 CA certificate. Supported beginning with 10.0.2. Used only when <code>s3Provider</code> is set to <code>custom</code> . Most of the custom S3 providers are created based on custom CA certificates. This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be <code>ca.crt</code> and the value is the base64-encoded value of the CA certificate. If the server certificate that is presented by your S3/object-store endpoint is signed by a well known CA and includes any intermediate certificates in the chain, then you do not need to provide the CA certificate using this feature because the Analytics subsystem already trusts the server certificate. You can test this by configuring the Analytics backup to your S3/object-store server without providing the CA certificate. If you do need to provide the CA certificate then you must provide the entire chain in the <code>ca.crt</code> member of the secret, as follows: <code>intermediate-certificate-1</code> <code>intermediate-certificate-2</code> <code>...</code> <code>intermediate-certificate-n</code> <code>root-certificate</code> Where the first certificate in the <code>ca.crt</code> member (<code>intermediate-certificate-1</code>) is the issuer of the S3/object-store server certificate, and each subsequent certificate in the <code>ca.crt</code> member is the issuer of the preceding certificate. The root certificate in the <code>ca.crt</code> member must be self-signed. If there are no intermediate certificates involved, then the <code>ca.crt</code> member contains only the root certificate. The following sample bash script creates the Kubernetes secret: <pre>cat >customs3ca.yaml <<EOF apiVersion: v1 data: ca.crt: \$(base64 <path-to-ca-certificate> tr -d '\n') kind: Secret metadata: name: custom-server-ca type: generic EOF kubectl apply -f customs3ca.yaml -n <namespace></pre>

Setting	Description
backups3URIS tyle	Indicates whether URIs to your S3 backup should use specify a host or a path value. Supported beginning with 10.0.2. Valid values: host and path . Used only when s3Provider is set to custom . Some custom S3 providers require URI style to be set to path . For example, MinIO supports both host and path style setup. You can create a MinIO S3 server to only accept path style client communications. Important: Contact your custom S3 administrator before configuring this field. If not properly configured, an upstream custom S3 can reject connections from the client, such as the API Connect Management subsystem.
host	For objstore type backup, specify S3 endpoint with the corresponding S3 region in the format <S3endpoint>/<S3region>
path	The path to the location of the backup: For objstore backups, the name of your S3 bucket to store backup data. For example, bucket-name/folder . The use of subdirectories in the bucket name is not supported. Note: When your deployment includes a management subsystem in two different clusters (with active databases), the two management subsystems cannot use the same s3 bucket name in the database backup configurations. Each management subsystem must use a unique s3 bucket name. Ensure that bucket-name/folder is empty. If the folder was previously used for backups, the folder will not be empty, and stanza create might encounter an error. Explanation: Once a folder is used, archive.info and backup.info files are created. During stanza creation, the process compares the database version and database system id between the two info files and the current Postgres database cluster. Stanza creation fails if there is a mismatch. To prevent this possibility, remove all files from the folder before configuring.
credentials	For objstore type backups, the name of the Kubernetes secret containing your S3 Access Key / Key Secret
schedule	Cron like schedule for performing automatic backups. The format for the schedule is: <ul style="list-style-type: none"> • ***** • ----- • • +----- day of week (0 - 6) (Sunday=0) • +----- month (1 - 12) • +----- day of month (1 - 31) • +----- hour (0 - 23) • +----- min (0 - 59) <p>The timezone for backups is that of the node on which the postgres-operator pod is scheduled.</p> <p>There is no default backup schedule set. Be sure to set your backup schedule.</p> <p>All scheduled Management subsystem backups are of type full only.</p>
repoRetentio nFull	v10.0.3.0 or later - The number of full S3 backups to retain. When the next full backup successfully completes, and the specified number of retained backups is reached, the oldest full backup is deleted. All incremental backups and archives associated with the oldest full backup also expire. Incremental backups are not counted for this setting. Applies to both manual backups and scheduled backups. Minimum value: 1 Maximum value: 999999 Default: none

3. Verify that the configuration deploys successfully.

When you configure Management subsystem backups, the operator runs a **stanza-create** job. This job creates the stanza (Postgres cluster configuration) on the upstream S3 server, which is used for backup and archive procedures. The job also brings up the necessary pod.

Check the status of the **stanza-create** job:

```
oc get jobs -n <namespace> | grep stanza
```

- On success:

- The **stanza-create** job status is 1/1:

```
oc get jobs | grep stanza
m1-b722e361-postgres-stanza-create      1/1      5s      127m
```

- The **stanza-create** job shows **status: "True"** and **type: Complete**:

```
oc get job m1-b722e361-postgres-stanza-create -o yaml
```

```
status:
  completionTime: "2021-04-13T18:38:50Z"
  conditions:
    - lastProbeTime: "2021-04-13T18:38:50Z"
      lastTransitionTime: "2021-04-13T18:38:50Z"
        status: "True"
        type: Complete
  startTime: "2021-04-13T18:38:45Z"
  succeeded: 1
```

- The **stanza-create** pod is in **Completed** state:

```
oc get pods | grep stanza
m1-b722e361-postgres-stanza-create-q4fvp 0/1      Completed 0      127m
```

- Exec into the pod:

```
oc exec -it <backrest-shared-repo-pod> -- bash
```


- The `pgbackrest` command returns the S3 contents in valid JSON:

```
pgbackrest info --output json --repol-type s3

{"backup":{"held":false}}, {"message":"ok"}}] [pgbackrest@m1-b722e361-postgres-backrest-shared-repo-69cdfdd5-rs882
/
```

- On failure, see [Troubleshooting the stanza-create job for S3 backup configuration on OpenShift and Cloud Pak for Integration](#).

Configuring SFTP backup settings for fresh install of the Management subsystem on OpenShift and Cloud Pak for Integration

Beginning with API Connect V10.0.1.1, you can configure SFTP backups for your Management subsystem in your OpenShift and Cloud Pak for Integration environments.

Before you begin

If the `APIConnectCluster` instance is already created, do not use these instructions. Instead, add the Management backup configuration by following the instructions in [Reconfiguring or adding backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration](#).

About this task

When creating an instance of API Connect, use the advanced configuration settings to configure database backups for the Management subsystem.

Procedure

1. Create a backup secret.

The backup secret is a Kubernetes secret that contains your credentials for accessing the SFTP backup database. Supported credentials types:

- Username and password (v10.0.2.0 or later).
- Username and SSH-key (v10.0.3.0 or later). Only OpenSSH keys are supported.¹

Use one of the following commands to create the secret:

- Username and password credentials

```
oc create secret generic mgmt-backup-secret --from-literal=username='<YOUR USERNAME>' --from-literal=password='<YOUR
PASSWORD>' -n <namespace-of-mgmt-subsystem>
```

- **Version 10.0.3.0 or later:** Username and SSH-key credentials:

```
oc create secret generic mgmt-backup-secret --from-literal=username='<YOUR USERNAME>' --from-file=ssh-
privatekey='<YOUR PRIVATEKEY FILE>' -n <namespace-of-mgmt-subsystem>
```

Note: If the username contains the backslash character, insert an extra backslash to escape it. For example, if the username is:

```
backup\svc-apic
```

set this as:

```
--from-literal=username='backup\\svc-apic'
```

2. Configure the Management backup settings.

Create the `APIConnectCluster` installation CR and add the Management backup configuration using one of the following methods:

- OpenShift web console:
 - Navigate to Advanced Configuration > ManagementSubsystem > Advanced Configuration > Database Backups > Advanced Configuration
 - See the on-screen instructions for how to fill in each field. See also [Management subsystem settings](#).
The following fields are unique to SFTP backup configuration:
 - Server port. Default is 22.
 - The fields in the Restart Database section. Changing SFTP (protocol) backup configures does not involve any downtime.
 - Retries.
- IBM Cloud Pak Platform UI:
 - In the Platform UI for the API Connect cluster instance, select Configuration. On the Common settings tab, set Advanced Options to On.
 - See the on-screen instructions for how to fill in values for:
 - Credentials
 - Server Hostname
 - Path
 - Server Port
 - Protocol
 - Retries
 - Schedule (The timezone for backups is that of the node on which the postgres-operator pod is scheduled.)

For more information, see [Management subsystem settings](#).
- YAML file: Use Table 1 for guidance while you define the backup settings as shown in the following example.

```
spec:
  management:
    databaseBackup:
      protocol: sftp
      host: <SFTP-host-name>
      port: <SFTP-port>
```

```

path: apic-backup
retries: 0
credentials: mgmt-backup-secret
schedule: "0 3 * * *"

```

Table 1. Backup configuration settings

Setting	Description
protocol	The type of the backup. For SFTP storage: sftp .
host	The backups host. For sftp type, the SFTP server hostname
port	The SFTP server port. Optional. Default: 22 .
path	The path to the location of the backup. For sftp type, the folder name on the SFTP server.
retries	The number of times the ibm-apiconnect Operator attempts backups in the event of a failed SFTP backup. Default value: 0 .
credentials	Name of the Kubernetes secret containing your SFTP Username/password. For Version 10.0.3.0 or later, the Kubernetes secret can contain your SFTP Username and SSH key.
schedule	<p>Cron like schedule for performing automatic backups. The format for the schedule is:</p> <ul style="list-style-type: none"> • ***** • ----- • • +----- day of week (0 - 6) (Sunday=0) • +----- month (1 - 12) • +----- day of month (1 - 31) • +----- hour (0 - 23) • +----- min (0 - 59) <p>The timezone for backups is that of the node on which the postgres-operator pod is scheduled.</p> <p>There is no default backup schedule set. Be sure to set your backup schedule.</p> <p>All scheduled Management subsystem backups are of type full only.</p>

¹ PuTTY style keys can be converted to OpenSSH by using the PuTTY Key Generator (PuTTYgen) application; see <https://www.puttygen.com/>.

Configuring local backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration

Configure local backups for the API Connect Management subsystem that you will deploy in your OpenShift or Cloud Pak for Integration environment.

About this task

When creating an instance of API Connect, use the advanced configuration settings to configure database backups for the Management subsystem.

Procedure

Configure the Management section of the top-level **APICoconnectCluster** installation CR with the **databaseBackup** subsection. Specify **schedule** and **repoRetentionFull**, the default settings are 01:00 UTC and 14 days.

```

databaseBackup:
  schedule: "0 1 * * *"
  repoRetentionFull: 14

```

Table 1. Backup configuration settings

Setting	Description
schedule	<p>Cron like schedule for performing automatic backups. The format for the schedule is:</p> <ul style="list-style-type: none"> • ***** • ----- • • +----- day of week (0 - 6) (Sunday=0) • +----- month (1 - 12) • +----- day of month (1 - 31) • +----- hour (0 - 23) • +----- min (0 - 59) <p>The timezone for backups is that of the node on which the postgres-operator pod is scheduled.</p> <p>The default is daily backups at 01:00 UTC.</p> <p>All scheduled Management subsystem backups are of type full only.</p>

Setting	Description
<code>repoRetentionFull</code>	<p>- The number of full local backups to retain. When the next full backup successfully completes, and the specified number of retained backups is reached, the oldest full backup is deleted. All incremental backups and archives associated with the oldest full backup also expire. Incremental backups are not counted for this setting. Applies to both manual backups and scheduled backups.</p> <p>Minimum value: 1</p> <p>Maximum value: 9999999</p> <p>Default: 14</p>

Reconfiguring or adding backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration

You can reconfigure the backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration.

About this task

Backup configurations under `management.databaseBackup` in `APIConnectCluster` instance can be modified using the OpenShift web console, IBM Cloud Pak Platform UI, or the command-line utility `oc`.

Note:

Using OpenShift web console or Platform UI can prevent updates because `APIConnectCluster` CR refreshes very quickly. If you encounter this problem, use the command line to modify backup configurations; for example:

```
oc edit apiconnectcluster <name-of-instance>
```

Procedure

1. Make sure that the management cluster status is `Running` and the `READY` condition displays the same value before and after the `"/`.

For example:

```
oc get mgmt
NAME      READY   STATUS    VERSION      RECONCILED VERSION  AGE
mgmt     16/16   Running   10.0.5-eus   10.0.5.1-eus-0-eus  19h
```

2. Create backup secret for your backup type. See the appropriate instructions:
 - [Configuring S3 backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration](#)
 - [Configuring SFTP backup settings for fresh install of the Management subsystem on OpenShift and Cloud Pak for Integration](#)
3. Complete the following steps, based on your storage type:
 - S3 storage

You can change or add the `DatabaseBackup` s3 configuration via the Management section of the API Connect Cluster CR after a fresh install. However, doing so requires a reconfiguration of the database and a brief downtime. The operator restarts the database with the new backup configurations as soon as the new configuration in the management CR is saved.

If you have a two data center disaster recovery deployment, before you configure S3 backup settings you must first disable 2DCDR on both sites. Follow these steps:

- Remove the `multiSiteHA` section from the `spec` section of the management CRs on both data centers. Take note of which data center is the active and which is the warm-standby. Keep a copy of the `multiSiteHA` sections that you remove.
- Configure your S3 backup settings. You must specify different backup locations for each data center, see [Backup and restore requirements for a two data center deployment](#).
- Add the `multiSiteHA` sections back to the management CRs in both data centers, ensuring that your original active is set to be the active again.

```
spec:
  management:
    databaseBackup:
      protocol: objstore
      s3provider: ibm
      host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
      path: apic-backup
      credentials: mgmt-backup-secret
      schedule: "0 3 * * *"
      restartDB:
        accept: true
```

Configuration of s3 backups uses a new parameter `restartDB` that is required when changing the backup configuration. `restartDB` is not required for initial deployments, only for modifications after the cluster is deployed.

Table 1. Additional parameter for S3 backup type reconfiguration

Parameter	Description
<code>restartDB:</code> <code>accept: <value></code>	<p>Required when changing the backup configuration. The parameter <code>accept</code> is a boolean: <code><value></code> can be <code>true</code> or <code>false</code>.</p> <p>When <code>accept</code> is <code>true</code>, and a new configuration for management CR is saved, the operator:</p> <ul style="list-style-type: none"> • Stops the database and any dependent services • Applies the change • Restarts the stopped components. <p>Note: <code>restartDB</code> is not required for initial deployments, only for modifications after the cluster is deployed.</p>

- SFTP

You can change/add the `DatabaseBackup` SFTP configuration via the Management section of the API Connect Cluster CR after a fresh install. This process does not involve any downtime. The same backup settings are used for reconfiguring as for initial configuration. For complete description of backup settings, see [Configuring SFTP backup settings for fresh install of the Management subsystem on OpenShift and Cloud Pak for Integration](#).

```
spec:
  management:
    databaseBackup:
      protocol: sftp
      host: <SFTP-hostname>
      port: <SFTP-port> (Optional, default is 22)
      path: apic-backup
      retries: 0
      credentials: mgmt-backup-secret
      schedule: "0 3 * * *"
```

Generating a manual backup of the management subsystem on OpenShift and Cloud Pak for Integration

You can generate a manual (non-scheduled) backup of the management subsystem in your OpenShift and Cloud Pak for Integration environments.

Procedure

1. Make sure that the management cluster status is Running and the READY condition displays matching values before and after the "\". You can use either the OpenShift web console or the `oc` command line to verify the health. For example:

```
oc get mgmt
NAME   READY   STATUS    VERSION      RECONCILED VERSION  AGE
mgmt   16/16   Running   10.0.1-eus   10.0.1.1-eus-0-eus  19h
```

2. Decide which of the two types of manual backups you want to do:
 - `full`: backs up the entire Management Subsystem database.
 - `incr`: backs up any data that has not been backed up since the last `full` or `incr` backup.
3. You generate a backup manually by creating a `managementBackup` custom resource which is detected by the `ibm-apiconnect` operator that then triggers the backup process.

You can create the `ManagementBackup` custom resource by using either the Form UI or the command line. For example:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementBackup
metadata:
  generateName: mgmt
spec:
  type: full
  crType: create
  clusterName: management
```

Table 1. CR settings for manual backup

Setting	Description
<code>type</code>	The type of backup to be done. Valid arguments are: <ul style="list-style-type: none"> • <code>full</code> Backs up the entire Management subsystem database • <code>incr</code> Backs up any data that has not been backed up since the last <code>full</code> or <code>incr</code> backup.
<code>crType</code>	Use <code>create</code> to trigger a backup process.
<code>clusterName</code>	The name of the target <code>ManagementCluster</code> you want to backup.

4. Create the `ManagementBackup` custom resource in the namespace of your Management Subsystem to trigger the backup process. Use the Form UI or the following command:

```
$ oc create -f mgmtbackup_cr.yaml -n <APIC_namespace>
```

5. (Optional): You can list current backups by using Form UI to list `ManagementBackup` resources, or by using `oc`:

```
$ oc get managementbackup -n <APIC_namespace> --sort-by=.metadata.creationTimestamp
```

Here is an example of some possible output:

```
NAME           STATUS    ID                CLUSTER  TYPE  CR TYPE  AGE
m1-e46bbac4    Ready    20201023-183005F  m1       full  record   66m
mgmt-backup-dnvxw  Ready    20201023-193137F  m1       full  create   9m37s
```

6. (Optional): You can get a detailed view of a specific backup by using the command:

```
$ oc get managementbackup <bup-name> -n <APIC_namespace> -o yaml
```

Here is an example of some possible output:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementBackup
metadata:
  creationTimestamp: "2020-10-23T18:34:11Z"
  generation: 1
  labels:
    app.kubernetes.io/instance: m1
    app.kubernetes.io/managed-by: ibm-apiconnect
```

```

  app.kubernetes.io/name: m1-e46bbac4
  name: m1-e46bbac4
  namespace: default
  resourceVersion: "6869"
  selfLink: /apis/management.apiconnect.ibm.com/v1beta1/namespaces/default/managementbackups/m1-e46bbac4
  uid: b7b6be9c-82d4-4737-984f-58584a3fe582
spec:
  clusterName: m1
  crType: create
  type: full
status:
  clusterName: m1-fbaa5be2-postgres
  conditions:
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupPending
    status: "False"
    type: Pending
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    message: Record of management backup complete
    reason: BackupComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupStatusUnknown
    status: "False"
    type: Warning
  crType: ""
  id: 20201023-183005F
  info:
    id: 20201023-183005F
    range: 2020-10-23 18:30:05 +0000 UTC - 2020-10-23 18:31:22 +0000 UTC
    rangeEnd: 1603477882
    rangeStart: 1603477805
    size: 32.6 MB
    totalsize: 32.6 MB
    type: full
  phase: Ready
  state: ""
  subsysName: m1

```

7. **Version 10.0.1.0 only:** If a local-backup was performed on v10.0.1.0, and you now wish to upload the local-backup file to an sftp server, see [Uploading backup files to an sftp server](#) for the required steps.

Restoring the management subsystem on OpenShift and Cloud Pak for Integration

You can restore your Management subsystem in your OpenShift and Cloud Pak for Integration environments.

Before you begin

You must have backups of the management subsystem in order to restore. See [Configuring backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration](#) and [Generating a manual backup of the management subsystem on OpenShift and Cloud Pak for Integration](#).

About this task

To trigger a restore, you will create a **ManagementRestore** custom resource. When the **ibm-apiconnect** operator detects the new CR, it starts the restore process.

- Ensure that the backup method that you used is complete before you attempt a restore. The backup data that is stored in the remote server is used for restoring the Management subsystem data back to a previous state. Ensure that these backups are on your remote server before you attempt a restore, and that the backup ID in the selected backup exists in remote storage.
- Restoring a backup restores the registration credentials (client_ID, client_secret) that were in use at the time that the selected backup was created. For information on the registration credentials, see [Changing the registration client id and client secret for applications](#).
- If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. The backups of the Management and Portal must be taken at the same time to ensure that the Portal sites are consistent with Management database.

Note: There is a known issue where sometimes a restore from the Management subsystem's SFTP backup succeeds but the data is not restored. If this happens, run the restore again.

Procedure

1. Use the Form UI or the **oc** command to obtain a list of the available backups, and decide which one to restore your Management subsystem:

```
oc get mgmtb -n <APIC_namespace>
```

For example:

```
$ oc get mgmtb
NAME                STATUS  ID                CLUSTER  TYPE  CR TYPE  AGE
```

management-3aa3bebf	Ready	20200929-152150F	management	full	record	10h
management-3c4ca8df	Ready	20200929-115520F	management	full	record	20h
management-5tbxj	Ready	20200929-224349F	management	full	create	17h
management-f10af337	Ready	20200925-133703F	management	full	record	5d2h
management-jtlcd	Ready	20200929-224349F	management	full	create	25h
management-zxjtz	Ready	20200929-224349F	management	full	create	28h

Attention: You can restore the Management subsystem from any backup where the STATUS is Ready; however, you should avoid restoring to the initial system backup. During installation, the Management database is used for an initial system backup before certain database schema jobs are complete. Restoring to this backup will result in an unstable system.

You will use one of the values in the **NAME** column.

2. Use the instructions that apply to your release of API Connect:

- Version 10.0.1.1 or later
 - Use the following code as an example:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementRestore
metadata:
  generateName: management-
spec:
  backupName: management-3aa3bebf
```

Note that **backupName**: is the name of the backup custom resource you want to restore to. For example, using the output from the previous step:

```
backupName: management-3aa3bebf
```

- Use the Form UI or **oc** to create the ManagementRestore custom resource in the namespace of the Management Subsystem, to trigger the restore process:

```
$ oc create -f mgmtrestore_cr.yaml -n <APIC_namespace>
```

- You can list current restores using the following command, or if using OCP UI you can list all **ManagementResource** list:

```
$ oc get managementrestore -n <APIC_namespace> --sort-by=.metadata.creationTimestamp
```

Example output:

- SFTP:

NAME	STATUS	BACKUP	CLUSTER	MESSAGE
PITR	AGE			
mgmt-restore-ptx6d	Complete	mgmt-backup-w4h8c	m1	Restore process completed (DB Restore +
DRR job)	6m31s			

- S3

NAME	STATUS	BACKUP	CLUSTER	MESSAGE
PITR	AGE			
mgmt-restore-hr9zx	Complete	mgmt-backup-xnjkl	m1	Restore process completed (DB Restore +
DRR job)	2020-11-23 23:08:32+00	9m8s		

- Version 10.0.1.0
 - Use the following command to obtain the Point-in-Time-Recovery target for the selected backup:

```
oc get mgmtb <backup_name> -o jsonpath="{..status.info.range}"
```

For example:

```
$ oc get mgmtb management-3aa3bebf -o jsonpath="{..status.info.range}"
2020-09-29 15:21:50 +0000 UTC - 2020-09-29 15:30:56 +0000 UTC
```

The target for recovery is the second half the **range** value in the backup custom resource. In the prior example: **2020-09-29 15:30:56 +0000**.

The custom resource uses the **pitrTarget** setting to specify the Point-in-Time-Recovery target. You must set **pitrTarget** to be later than the value obtained above (such as by one second newer). Use the format **YYYY-MM-DD HH:MM:SS+Z**.

Using the example above, where **2020-09-29 15:30:56 +0000** is the **range**, the **pitrTarget** is **2020-09-29 15:30:57+00:**

```
pitrTarget: "2020-09-29 15:30:57+00"
```

- Use the following code as an example:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementRestore
metadata:
  generateName: management-
spec:
  backupName: management-3aa3bebf
  pitrTarget: "2020-09-30 13:53:21+00" ## Not required in 10.0.1.1-eus and greater
```

- **backupName** - The name of the backup custom resource you want to restore to. For example, using the output from the previous step, **management-3aa3bebf**.
- **pitrTarget**: - The time you want to restore to, in format **YYYY-MM-DD HH:MM:SS+Z**.
- Use the Form UI or **oc** to create the ManagementRestore custom resource in the namespace of the Management Subsystem, to trigger the restore process:

```
$ oc create -f mgmtrestore_cr.yaml -n <APIC_namespace>
```

- You can list current restores using the following command, or if using Form UI you can list all **ManagementResource** lists:

```
$ oc get managementrestore -n <APIC_namespace> --sort-by=.metadata.creationTimestamp
```

Example output:


NAME	STATUS	BACKUP	CLUSTER	PITR	AGE
management-6nzjr	Complete	mgmt-backup-dg4s4	m1	2020-01-05 19:05:00+01	14m

- The operator creates an upgrade CR when the postgres database is restarted during the restore:

```
managementdbupgrade.management.apiconnect.ibm.com/m1-up-zqkk5 Complete Fresh install is Complete (DB Schema/data are up-to-date) 63s5s
```

Troubleshooting management subsystem backups on OpenShift and Cloud Pak for Integration

You can troubleshoot failures of the API Connect management subsystem backups in an OpenShift or Cloud Pak for Integration environment.

Note: In the  Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
See:

- [Troubleshooting resiliency issues on OpenShift and Cloud Pak for Integration](#)
Resolve resiliency issues with API Connect management system backups.
- [Troubleshooting the stanza-create job for S3 backup configuration on OpenShift and Cloud Pak for Integration](#)
Troubleshoot configuration of s3 backups when the stanza-create job fails in an OpenShift or Cloud Pak for Integration environment.

Troubleshooting resiliency issues on OpenShift and Cloud Pak for Integration

Resolve resiliency issues with API Connect management system backups.

pgbackrest-shared-repo resiliency problems

If the Kubernetes node hosting `pgbackrest-shared-repo` pod is down and the PVC attached to the pod has strict zone requirements (for example, in AWS or other clouds) or if the storage class is set to `local-storage`, then `pgbackrest-shared-repo` pod will not get rescheduled to another Kubernetes node.

As a result, there is a single point of failure and the following conditions might occur:

- Backups of the management database fails
- Disk space fills with accumulated Postgres wal (Write-Ahead Logging) files

To avoid this problem, monitor the disk usage, see: [Monitoring Postgres disk usage on OpenShift](#) or [Monitoring Postgres disk usage on Cloud Pak for Integration](#).

When the Kubernetes node comes back up, the pod is scheduled and all the processes should resume properly.

Troubleshooting the stanza-create job for S3 backup configuration on OpenShift and Cloud Pak for Integration

Troubleshoot configuration of s3 backups when the stanza-create job fails in an OpenShift or Cloud Pak for Integration environment.

When you configure Management subsystem backups for s3 providers, the operator runs a `stanza-create` job. This job creates the stanza (Postgres cluster configuration) on the upstream S3 server, which is used for backup and archive procedures. The job also brings up the necessary pod.

Check the status of the `stanza-create` job:

```
oc get jobs -n <namespace> | grep stanza
```

On failure:

- `stanza-create` pods fail, and job status is 0/1:

```
oc get jobs | grep stanza
m1-3bfe12ac-postgres-stanza-create 0/1 32m 32m
```

- Listing all pods brought up by `stanza-create` job shows multiple pods because a job tries `backoffLimit` times to complete the job. By default, there will be 7 pods in error state:

```
oc get pods | grep stanza
m1-3bfe12ac-postgres-stanza-create-726z4 0/1 Error 0 11m
m1-3bfe12ac-postgres-stanza-create-9vq4n 0/1 Error 0 22m
m1-3bfe12ac-postgres-stanza-create-gbtgb 0/1 Error 0 30m
m1-3bfe12ac-postgres-stanza-create-g15g5 0/1 Error 0 17m
m1-3bfe12ac-postgres-stanza-create-q8qhw 0/1 Error 0 26m
m1-3bfe12ac-postgres-stanza-create-t84zs 0/1 Error 0 8m54s
m1-3bfe12ac-postgres-stanza-create-z8tfs 0/1 Error 0 12m
```

Note: In some cases, the pods are cleaned by Kubernetes and the logs of these pods can be lost.

- Pod log errors show a reason for the failure. For example, when a hostname is not valid:

```
kubectl logs m1-3bfe12ac-postgres-stanza-create-gbtgb
```

```
time="2021-04-13T20:01:19Z" level=info msg="pgo-backrest starts"
time="2021-04-13T20:01:19Z" level=info msg="debug flag set to false"
time="2021-04-13T20:01:19Z" level=info msg="backrest stanza-create command requested"
time="2021-04-13T20:01:19Z" level=info msg="s3 flag enabled for backrest command"
time="2021-04-13T20:01:19Z" level=info msg="command to execute is [pgbackrest stanza-create --db-host=192.1.2.3 --db-path=/pgdata/m1-3bfe12ac-postgres --repol-type=s3]"
time="2021-04-13T20:01:19Z" level=info msg="command is pgbackrest stanza-create --db-host=192.1.2.3 --db-path=/pgdata/m1-3bfe12ac-postgres --repol-type=s3 "
time="2021-04-13T20:05:21Z" level=error msg="command terminated with exit code 49"
time="2021-04-13T20:05:21Z" level=info msg="output=[]"
```

- Job status is set to `type: Failed` and `status: "True"`, with `reason: BackoffLimitExceeded`:

```
oc get job m1-3bfe12ac-postgres-stanza-create -o yaml
```

```
status:
  conditions:
  - lastProbeTime: "2021-04-13T20:28:03Z"
    lastTransitionTime: "2021-04-13T20:28:03Z"
    message: Job has reached the specified backoff limit
    reason: BackoffLimitExceeded
    status: "True"
    type: Failed
    failed: 7
  startTime: "2021-04-13T20:01:18Z"
```

- If pod logs are lost, you can execute a command inside the `backrest-shared-repo` pod:
 - Obtain the `backrest-shared-repo` pod name:

```
oc get pods -n <namespace> | grep backrest-shared-repo
```

For example:

```
oc get pods | grep backrest-shared-repo
m1-eb8edc18-postgres-backrest-shared-repo-98dd46cc6-tw195 1/1 Running
```

- Exec into the pod:

```
oc exec -it <backrest-shared-repo-pod> -- bash
```

- Run:

```
pgbackrest info --output json --repol-type s3
```

For example, for invalid hostname:

```
ERROR: [049]: unable to get address for 'test1-v10-backup.s3.exampleaws.com': [-2] Name or service not knownCopy
```

The exact **ERROR**: will vary depending on the misconfiguration. Common errors:

```
Invalid access key
Invalid access key secret
Invalid bucket region
Invalid bucket or folder path
```

- To fix the incorrect settings, see [Reconfiguring or adding backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration](#).

Backing up and restoring Developer Portal on OpenShift and Cloud Pak for Integration

You can back up and restore your Developer Portal service in your OpenShift or Cloud Pak for Integration environment.

About this task

Before you can initiate a backup or restore operation, you must update the Portal CR (custom resource) to configure your settings. Generating a backup for Portal is performed on demand by deploying a backup CR. You can deploy a restore CR to start the restoration of a previously generated backup.

For points to consider when you are backing up and restoring a two data center disaster recovery deployment, see [Backup and restore requirements for a two data center deployment](#).

The default Developer Portal backup schedule is once every 24 hours, but the schedule can be changed in the backup settings. The Developer Portal saves all system and site backups locally, and also saves them remotely based on the configured SFTP and s3 settings.

The local backups are automatically maintained so that the latest three backups of each site and of the system are kept, and older backups are removed. This maintenance means that the Developer Portal retains the latest three backups for each site and for the system however old they are, but there is no deletion of the old backups on the remote server. If a site is deleted, then all of the local backups for that site are also deleted, as otherwise the backup volume might become full of old site backups. For remote backups, you can configure a retention policy on your remote server to remove the old backup files as required.

Important: Backups of the Management and Portal subsystems must be taken at the same time to ensure that the Portal sites are consistent with Management database. When you perform a restore, you must complete the restoration of the Management subsystem first, and then immediately restore the Developer Portal subsystem.

- [Configuring backups for Developer Portal on OpenShift and Cloud Pak for Integration](#)
Configure backups for the Developer Portal service in your OpenShift or Cloud Pak for Integration environment.
- [Backing up Developer Portal on OpenShift and Cloud Pak for Integration](#)
Perform a manual backup of the Developer Portal subsystem in your OpenShift or Cloud Pak for Integration environment.
- [Restoring Developer Portal on OpenShift and Cloud Pak for Integration](#)
You can restore the Developer Portal subsystem and sites in your OpenShift or Cloud Pak for Integration environment.
- [Troubleshooting Developer Portal backup and restore on OpenShift and Cloud Pak for Integration](#)
Review details on a failed backup or restore operation for Developer Portal backup on OpenShift and Cloud Pak for Integration.

Configuring backups for Developer Portal on OpenShift and Cloud Pak for Integration

Configure backups for the Developer Portal service in your OpenShift or Cloud Pak for Integration environment.

About this task

Run commands in the namespace where the Developer Portal is installed.

For points to consider when you are backing up and restoring a two data center configuration, see [Backup and restore requirements for a two data center deployment](#).

The default Developer Portal backup schedule is once every 24 hours, but the schedule can be changed in the backup settings. The Developer Portal saves all system and site backups locally, and also saves them remotely based on the configured SFTP and s3 settings.

The local backups are automatically maintained so that the latest three backups of each site and of the system are kept, and older backups are removed. This maintenance means that the Developer Portal retains the latest three backups for each site and for the system however old they are, but there is no deletion of the old backups on the remote server. If a site is deleted, then all of the local backups for that site are also deleted, as otherwise the backup volume might become full of old site backups. For remote backups, you can configure a retention policy on your remote server to remove the old backup files as required.

Procedure

1. Create your Developer Portal backup secret.

The backup secret is a Kubernetes secret that contains your credentials for accessing the s3 or sftp backup database.

- S3
Only password-based authentication is supported for S3, not authentication based on public certificates and private keys. Password-based authentication for S3 requires that you generate an access key and secret. For example:
 - IBM (Cloud Object Storage): [Service credentials](#).
 - AWS: [Managing access keys](#).

The secret can be created with the following command:

```
oc create secret generic portal-backup-secret --from-literal=username='Your_access_key-or-user_name' --from-literal=password='Your_access_key_secret-or-password' -n <Portal_namespace>
```

- SFTP
Supported credentials types:
 - User name and password (v10.0.2.0 or later).
 - User name and SSH-key (v10.0.3.0 or later). Only OpenSSH keys are supported.¹

Use one of the following commands to create the secret:

- User name and password credentials

```
oc create secret generic portal-backup-secret --from-literal=username='<Your_user_name>' --from-literal=password='<Your_password>' -n <Portal_namespace>
```

- Version 10.0.3.0 or later: User name and SSH-key credentials:

```
oc create secret generic portal-backup-secret --from-literal=username='<Your_user_name>' --from-file=ssh-privatekey='<Your_private_key_file>' -n <Portal_namespace>
```

2. Add a `portalBackup` subsection to the `spec.portal` section of the top-level `apiconnectcluster` CR.

You can edit the YAML in the UI, or you can run the following command and then add the `portalBackup` block:

```
oc edit apiconnectcluster <Portal_namespace>
```

An example when you use `protocol: objstore`:

```
spec:
  portal:
    portalBackup:
      credentials: portal-backup-secret
      host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
      path: test-bucket/restore-test
      port: 443
      backupCerts: <custom-s3-server-CA-cert>
      backups3URISyle: <host-or-path>
      schedule: '0 2 * * *'
```

An example when you use `protocol: sftp`:

```
spec:
  portal:
    portalBackup:
      credentials: portal-backup-secret
      host: sftp-service.example.com
      path: /home/fvtuser/site-backups
      port: 22
      protocol: sftp
      schedule: '0 2 * * *'
```

Notes about the `portalBackup` entry:

- If `portalBackup` entry is empty or omitted from the API Connect Cluster CR, then `local` backups will be taken if a backup CR is created. If the `schedule` exists, backups are taken at the scheduled time.
- For S3 and SFTP, the `portalBackup` entry must contain host, credentials, protocol, and path.
- For S3, the `portalBackup` section can optionally contain either, or both, of `backupCerts` and `backupS3URISyle`.

Table 1. Developer Portal backup settings

Setting	Description
<code>credentials</code>	<p>The name of the secret you created. Supported credential types:</p> <ul style="list-style-type: none"> • S3 Only password-based authentication is supported for S3, not authentication based on public certificates and private keys. Password-based authentication for S3 requires that you generate an access key and secret. • SFTP <ul style="list-style-type: none"> • User name and password (v10.0.2.0 or later) • User name and SSH-key (v10.0.3.0 or later)
<code>host</code>	<ul style="list-style-type: none"> • S3 - <code>host</code>: is the S3 endpoint with the corresponding S3 region in the format <code><S3endpoint></code>, or with an optional S3 region in the format <code><S3endpoint>/<S3region></code>. • SFTP - the hostname of the sftp server
<code>path</code>	<ul style="list-style-type: none"> • S3 - the name of your S3 bucket to store backup data, optionally followed by a / and then any subdirectories inside the bucket. Valid examples: <pre>bucket bucket/path bucket/lots/of/paths</pre> • SFTP - the full absolute path of the folder on the SFTP server, beginning with /. <p>Attention: Ensure that your backup <code>path</code> has the correct permissions to allow your backup user to be able to read and write to it from within the Developer Portal.</p>
<code>port</code>	<p>The port for the protocol to connect to the host.</p> <ul style="list-style-type: none"> • S3 - Ignored for S3 (because objstore always connects to port 443) • SFTP - Defaults to 22 if not specified • local - Ignored for local
<code>protocol</code>	<p>The protocol that is used to communicate with your remote backup endpoint. Specify one of the following values:</p> <ul style="list-style-type: none"> • <code>sftp</code> - for secure file transfer protocol. • <code>objstore</code> - for S3 compatible object storage.

Setting	Description
backupCerts	<p>(S3 storage only) backupCerts is the name of a custom certificate that contains your upstream custom S3 CA certificate. Supported beginning with 10.0.2.</p> <p>Note: If you use a public certificate for the S3 storage provider, it must be signed by a known certificate authority that is trusted by API Connect. Use of an untrusted authority can cause the following error during backup upload:</p> <pre>x509: certificate signed by unknown authority</pre> <p>This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be <code>ca.crt</code> and the value is the base64-encoded value of the CA certificate.</p> <p>If the server certificate that is presented by your S3/object-store endpoint is signed by a well known CA and includes any intermediate certificates in the chain, then you do not need to provide the CA certificate using this feature because the Portal subsystem already trusts the server certificate. You can test this by configuring the Portal backup to your S3/object-store server without providing the CA certificate.</p> <p>If you do need to provide the CA certificate then you must provide the entire chain in the <code>ca.crt</code> member of the secret, as follows:</p> <pre>intermediate-certificate-1 intermediate-certificate-2 . . intermediate-certificate-n root-certificate</pre> <p>Where the first certificate in the <code>ca.crt</code> member (intermediate-certificate-1) is the issuer of the S3/object-store server certificate, and each subsequent certificate in the <code>ca.crt</code> member is the issuer of the preceding certificate. The root certificate in the <code>ca.crt</code> member must be self-signed.</p> <p>If there are no intermediate certificates involved, then the <code>ca.crt</code> member contains only the root certificate.</p> <p>The following sample bash script creates the Kubernetes secret:</p> <pre>cat >customs3ca.yaml <<EOF apiVersion: v1 data: ca.crt: \$(base64 <path-to-ca-certificate> tr -d '\n') kind: Secret metadata: name: custom-server-ca type: generic EOF kubectl apply -f customs3ca.yaml -n <APIC_namespace></pre>
backups3URIStyle	<p>(S3 storage only) Optional - backups3URIStyle indicates whether URIs to your S3 backup should use specify a host or a path value. When not specified, default to <code>host</code>. Supported beginning with 10.0.2.</p> <p>Valid values: <code>host</code> and <code>path</code>.</p> <p>Some custom S3 providers require URI style to be set to <code>path</code>. For example, MinIO supports both <code>host</code> and <code>path</code> style setup. You can create a MinIO S3 server to only accept <code>path</code> style client communications.</p> <p>Important: Contact your custom S3 administrator before configuring this field. If not properly configured, an upstream custom S3 can reject connections from the client, such as the API Connect Management subsystem.</p>
schedule	<p>The schedule for how often automatic Developer Portal backups are run. The format for the schedule is any valid cron string. The timezone for backups is UTC.</p> <p>Valid for all backup types: S3, SFTP, and local.</p> <pre>spec: portal: portalBackup: . . schedule: '0 2 * * *'</pre>

- [Overview of the Developer Portal backup resources](#)
How to manage your Developer Portal remote backup resources.

¹ PuTTY style keys can be converted to OpenSSH by using the PuTTY Key Generator (PuTTYgen) application; see <https://www.puttygen.com/>.

Overview of the Developer Portal backup resources

How to manage your Developer Portal remote backup resources.

Note: This information is relevant only if you're taking remote Developer Portal backups.

The Developer Portal uses two types of portal backup resources. When you run the command `kubectl get portalbackup`, you see a list of backups, for example:

NAME	ID	TASK ID	STATUS	TYPE	CR TYPE	CLUSTER	AGE	COMMENT
my-backup-f91ms	n/a	20b47fe090f5	Ready	system	create	portal	78m	
portal-record-222cg	20220108.010033		Ready	system	record	portal	94m	
portal-record-22xdc	20220702.010339		Ready	site	record	portal	9m49s	
portal-record-245j7	20221002.010340		Ready	site	record	portal	5m	
portal-record-246hm	20221218.010103		Ready	site	record	portal	32m	
portal-record-24h44	20220509.010037		Ready	site	record	portal	68m	
portal-record-24wb4	20221003.010029		Ready	site	record	portal	60m	
portal-record-264mh	20221119.010453		Ready	system	record	portal	78m	

There are two types of resources listed, and these are denoted by the **CR TYPE** column; **create** or **record** types.

1. **create**: This resource is a representation of a **portalbackup** task that was created manually. See [Backing up and restoring the Developer Portal in a Kubernetes environment](#), [Backing up Developer Portal on OpenShift and Cloud Pak for Integration](#), or [Backing up the Developer Portal subsystem](#) on VMware, for information about how to create a manual portal backup.
2. **record**: This resource is generated by the operator, and it's a representation of a backup file that's stored on the remote backup server. A record listing can be modified only to add a comment to it. You can describe each record to understand which file it is referring to by running the following command:

```
kubectl describe portalbackup portal-record-name
```

For example:

```
kubectl describe portalbackup portal-record-zmmfw
.
Name:          portal-record-zmmfw
Namespace:     apic
Labels:        createdBy
.
Full Backup Name:  my-portal-s3-hostname@my-porg@catal-20220604.010302.tar.gz
```

Record rows are generated and updated on a schedule, according to the backups that exist on your remote backup server. If you have 10 backup files stored on your remote server, then there will be 10 record rows; one row for each file.

The Developer Portal doesn't have a retention policy for remote backups. Therefore, it's your responsibility to ensure that you're keeping the number of backups that are stored remotely under control. An excessive number of portal backups results in delays administrating and upgrading the Developer Portal. This is due to the operator needing to maintain the excessive number of record rows that correspond to each remote portal backup.

Backing up Developer Portal on OpenShift and Cloud Pak for Integration

Perform a manual backup of the Developer Portal subsystem in your OpenShift or Cloud Pak for Integration environment.

About this task

Run commands in the namespace where the Developer Portal is installed.

Important: Backups of the Management and Portal subsystems must be taken at the same time to ensure that the Portal sites are consistent with Management database. When you perform a restore, you must complete the restoration of the Management subsystem first, and then immediately restore the Developer Portal subsystem. The default Developer Portal backup schedule is once every 24 hours, but the schedule can be changed in the backup settings. The Developer Portal saves all system and site backups locally, and also saves them remotely based on the configured SFTP and s3 settings.

The local backups are automatically maintained so that the latest three backups of each site and of the system are kept, and older backups are removed. This maintenance means that the Developer Portal retains the latest three backups for each site and for the system however old they are, but there is no deletion of the old backups on the remote server. If a site is deleted, then all of the local backups for that site are also deleted, as otherwise the backup volume might become full of old site backups. For remote backups, you can configure a retention policy on your remote server to remove the old backup files as required.

Tip: If you are using remote backups, and there are a large number of rows in your **portalbackup** list that have a **CR TYPE** of **record**, there might be a slow down in system performance, including during Developer Portal upgrades. Although there is no upper limit to how many portal backups you can have, it is recommended that you prune your remote backups so that the number of rows does not become excessively large. One **portalbackup** row of type **record**, corresponds to one backup file on the configured remote backup server. For more information, see [Overview of the Developer Portal backup resources](#).

Procedure

1. Required: Backup the Management subsystem as explained in [Generating a manual backup of the management subsystem on OpenShift and Cloud Pak for Integration](#).
2. Create the **PortalBackup** CR (custom resource) as a YAML file with the backup options that you want to use for Developer Portal. You can use `portalbackup_cr.yaml` in the `helper_files` directory as an example:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalBackup
metadata:
  generateName: portal-bup-
spec:
  crType: create
  portalCluster: portal
  type: all
  comment: "test comment"
```

Set the following backup properties as needed:

- **crType** - Use **create** to trigger a backup process. When the backup is created, the operator generates another backup CR with **crType**: **record**, which represents the stored backup that you can use for a restore operation.
- **portalCluster** - the name of the portal cluster you want to backup. You can find this with:

```
oc get portalcluster -n <namespace>
```

- **record** - Includes a backup ID so you can choose it for restoring. If you plan to preserve the backup for use in recovering from a disaster, use the **record** argument to ensure that all necessary data is captured.
- **type** - The extent of the backup; select one of the following options:
 - **all** - Backup the Developer Portal subsystem and all of the sites
 - **system** - Backup the Developer Portal subsystem

- **site** - Backup only a specified Developer Portal site
 - **siteName**: Used only when you perform a **site** backup. Valid arguments are:
 - **site uuid** or **url** - back up only the specified site
 - **installed** - backup all installed sites
 - **comment**: Optional. You can optionally add a comment (such as a description or special notes) to the CR before you create a backup.
3. Run the following command to create the backup CR, which triggers the backup operation:

```
oc create -f portalbackup_cr.yaml -n <APIC_namespace>
```

4. Verify that the backup was successful by completing the following steps:
- Run the following command to get a list of backups:

```
oc get portalbackup -n <APIC_namespace>
```

The response looks like the following example, where only **crType**: **record** backups include an ID.

NAME	ID	STATUS	TYPE	CR TYPE	AGE	COMMENT
p1-bup-gqz8f	20200526.142449	Ready	system	record	55m	
p1-bup-jx4vc	20200526.142458	Ready	site	record	54m	
p1-bup-st412		Ready	all	create	56m	y
p1-bup-th2dm		Ready	all	create	4h51m	y

- Run the following command to view the details of a particular backup,, which is specified using the **<bup-name>** from the **NAME** column in the list of backups:

```
oc get portalbackup <bup-name> -n <APIC_namespace> -o yaml
```

The details of a backup look like the following example:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalBackup
metadata:
  creationTimestamp: "2020-10-01T10:00:16Z"
  generateName: portal-bup-
  generation: 1
  name: portal-bup-g66gv
  namespace: default
  resourceVersion: "8940"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/default/portalbackups/portal-bup-g66gv
  uid: b1d5b295-b966-48f2-b6ef-dfcd1c499741
spec:
  comment: test comment
  portalCluster: portal
  crType: create
  siteName: installed
  type: site
status:
  backupId: ""
  commentLeft: "y"
  conditions:
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    message: |-
      2020-10-01 10:00:18: CLI task (backup:site) starting.
      2020-10-01 10:00:20: Making a backup for site conor.com
      2020-10-01 10:00:27: A local backup was successfully created: /var/aegir/backups/conor.com-20201001.100021.tar.gz
      2020-10-01 10:00:30: Attempting to upload backup to remote backup server
      2020-10-01 10:00:30: Uploading conor.com-20201001.100021.tar.gz to s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard:22:cq-test/ptlcq4/ using objstore
      2020-10-01 10:00:33: Site with URL conor.com (conor.com-20201001.100021.tar.gz) successfully uploaded
      2020-10-01 10:00:33: All installed sites have been successfully backed up
      2020-10-01 10:00:33: CLI task (backup:site) completed successfully.
    reason: BackupComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    reason: BackupInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupStatusUnknown
    status: "False"
    type: Warning
  fullBackupName: ""
  id: f8t0ux7z4zfnjnu
  message: ""
  phase: Ready
  state: ""
```

- Check the contents of the backup and make sure that it contains the following statements, which indicate a successful operation:

```
. . .
  reason: BackupComplete
  status: "True"
  type: Ready
```

If the operation did not complete successfully, see [Troubleshooting Developer Portal backup and restore on OpenShift and Cloud Pak for Integration](#) for information on reviewing logs to help resolve the problem.

Restoring Developer Portal on OpenShift and Cloud Pak for Integration

You can restore the Developer Portal subsystem and sites in your OpenShift or Cloud Pak for Integration environment.

About this task

Run commands in the namespace where the Developer Portal is installed.

Important: Backups of the Management and Portal subsystems must be taken at the same time to ensure that the Portal sites are consistent with Management database. When you perform a restore, you must complete the restoration of the Management subsystem first, and then immediately restore the Developer Portal subsystem.

Procedure

1. Required: Restore up the Management subsystem as explained in [Generating a manual backup of the management subsystem on OpenShift and Cloud Pak for Integration](#).
2. Choose which backup you want to restore.
 - a. Run the following command to get a list of available backups:

```
oc get portalbackup -n <APIC_namespace>
```

The response looks like the following example:

NAME	ID	STATUS	TYPE	CR TYPE	AGE	COMMENT
p1-bup-gqz8f	20200526.142449	Ready	system	record	55m	
p1-bup-jx4vc	20200526.142458	Ready	site	record	54m	
p1-bup-st412		Ready	all	create	56m	y
p1-bup-th2dm		Ready	all	create	4h51m	y

Note: You can only restore the Portal subsystem from backups where the **CR TYPE** is **record**.

- b. Run the following command to view the details of a particular backup, which is specified using the `<bup-name>` from the **NAME** column in the list of backups:

```
oc get portalbackup <bup-name> -n <APIC_namespace> -o yaml
```

The details of a backup look like the following example:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalBackup
metadata:
  creationTimestamp: "2020-10-01T10:00:16Z"
  generateName: portal-bup-
  generation: 1
  name: portal-bup-g66gv
  namespace: default
  resourceVersion: "8940"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/default/portalbackups/portal-bup-g66gv
  uid: b1d5b295-b966-48f2-b6ef-dfcd1c499741
spec:
  comment: test comment
  portalCluster: portal
  crType: record
  siteName: installed
  type: site
status:
  backupId: ""
  commentLeft: "y"
  conditions:
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    message: |-
      2020-10-01 10:00:18: CLI task (backup:site) starting.
      2020-10-01 10:00:20: Making a backup for site conor.com
      2020-10-01 10:00:27: A local backup was succesfully created: /var/aegir/backups/conor.com-20201001.100021.tar.gz
      2020-10-01 10:00:30: Attempting to upload backup to remote backup server
      2020-10-01 10:00:30: Uploading conor.com-20201001.100021.tar.gz to s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard:22:cq-test/ptlcq4/ using objstore
      2020-10-01 10:00:33: Site with URL conor.com (conor.com-20201001.100021.tar.gz) successfully uploaded
      2020-10-01 10:00:33: All installed sites have been successfully backed up
      2020-10-01 10:00:33: CLI task (backup:site) completed successfully.
    reason: BackupComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    reason: BackupInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupStatusUnknown
    status: "False"
    type: Warning
  fullBackupName: ""
  id: f8t0ux7z4zfnjnju
  message: ""
```

```

phase: Ready
state: ""

```

3. Create the `PortalRestore` CR (custom resource) as a YAML file with the backup options that you want to use for Developer Portal. You can use `portalrestore_cr.yaml` in the `helper_files` directory as an example:

```

apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalRestore
metadata:
  generateName: portal-restore-
spec:
  type: all
  portalCluster: portal
  dryRun: true
  timestamp: now
  priorityList:
    - myportal-critical-site.com
    - myportal-important.com

```

Add the following restore options as needed. Table 1 indicates which options are needed for each type of restore operation:

Table 1. The options that are needed for each type of Developer Portal restore operation

Option	Applies to all	Applies to system	Applies to site
type	X	X	X
portalCluster	X	X	X
siteName			X
systemName		X	
timestamp	X		
dryRun	X		
priorityList			

- **type**: Indicates the extent of the restore operation; specify one of the following options:
 - **all** - Restore the Developer Portal subsystem and all of the sites (supports only remote restores).
 - **system** - Restore the Developer Portal subsystem.
 - **site** - Restore only a specified Developer Portal site (you will indicate the site with the `siteName` option).
- **portalCluster** is the name of the portal cluster you want to restore. You can find this with:


```
oc get portal -n <namespace>
```
- **siteName**: Indicates which site to restore (used only with the `site` option); specify one of the following options:
 - **ID** - Restore the backup specified by the ID. The ID must correspond to a Developer Portal backup of `type: site` and `crType: record`.
 - **site-1234.tgz** - Restore the site by using this file name (which must exist on the local file system or remote server).
 - **myportal.com**: find the latest backup for this URL (locally or remotely) and restore it.
 - **all** - Restore all sites that are found on the remote server by using the latest backup.
 - **site** - Restore only the site that matches the specified uuid or url.
 - **installed** - Restore all installed sites.
- **systemName** - Restore from a backup that corresponds to the specified system; use one of the following options:
 - **ID** - Restore the backup specified by the ID. The ID must correspond to a Developer Portal backup of `type: site` and `crType: record`.
 - **system_backup-1234.tgz** - Restore the system from the given file that exists on the pod's local file system or the remote server.
 - **latest** - Restore from latest subsystem backup that is found on either the local file system or the remote server.
- **timestamp** - Valid arguments are:
 - **now** - Restore using the latest backups available.
 - **< TIMESTAMP >** - Specified in `'YYYYMMDD.HHMMSS'` format to indicate a timestamp to retrieve the backup from. The nearest backup, searching backwards from this timestamp, is used.

Note: From IBM® API Connect Version 10.0.3.0, the timestamp format changed to `YYYYMMDD.HHMMSS`. For Version 10.0.2.0 and earlier, the timestamp format is `YYYY-MM-DD HH:MM:SS`.
- **dryRun**: Indicates whether an actual restore operation is run; use one of the following options:
 - **true** - Execute a test, or dry run, of the command and return the list of actions that will be taken. These actions can be found in the custom resource.
 - **false** - Run a real restore operation, replacing the subsystem files and all sites. Any existing sites are reinstalled.
- **priorityList** - Restore sites in the sequence in which they are listed. The priority sites are queued for restoring first, followed by any remaining sites present in the backup location. Valid arguments are:
 - **myportalsite.com**
 - **mysecondportalsite.com**
- **customPlatformApiHostname** - If the API hostname of your management subsystem platform has changed since taking the backup you are restoring, then update it to the new platform API hostname of the management subsystem. (Optional)

4. Run the following command to create the restore CR, which triggers the restore operation:

```
oc create -f portalrestore_cr.yaml -n <APIC_namespace>
```

5. When the restore operation is finished, verify that the restore was successful by completing the following steps:

- a. Run the following command to get the name of the most recent restore record:

```
oc get portalrestore -n <APIC_namespace>
```

The response looks like the following example:

```

NAME                READY   AGE
portal-restore-twjrp Ready   28s

```

- b. Run the following command to view information about a specific restore operation, which is indicated with the `<portal_restore_name>` shown in the `NAME` column:

```
oc get portalrestore <portal_restore_name> -n <APIC_namespace> -o yaml
```

The details of a restore look like the following example:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalRestore
metadata:
  creationTimestamp: "2020-11-11T10:14:39Z"
  generateName: portal-restore-
  generation: 1
  name: portal-restore-8f4c5
  namespace: portal
  resourceVersion: "1497597"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/portal/portalrestores/portal-restore-8f4c5
  uid: 5a70b657-d998-4989-b9b6-8480e986311d
spec:
  siteName: "20200526.090209"
  portalCluster: portal
  type: site
status:
  backupId: ""
  commentLeft: ""
  conditions:
  - lastTransitionTime: "2020-11-11T10:14:39Z"
    reason: RestoreFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-11-11T10:14:39Z"
    reason: RestorePending
    status: "False"
    type: Pending
  - lastTransitionTime: "2020-11-11T14:55:32Z"
    message: |-
      2020-05-26 13:11:25: CLI task (restore:site) starting.
      2020-05-26 13:11:36: Deleting any old site data if it exists...
      2020-05-26 13:13:18: Restoring site test.com ...
      2020-05-26 13:13:18: .
      2020-05-26 13:13:19: Patching the database dump before restoring ...
      2020-05-26 13:13:48: .
      2020-05-26 13:19:41: Triggering the drush command to restore the site. This may take some time.
      2020-05-26 13:19:48: .
      2020-05-26 13:25:40: Drush restore command completed. Setting up the site.
      2020-05-26 13:25:48: .
      2020-05-26 13:30:49: Site test.com restored from backup /var/aegir/backups/test.com-20200526.090209.tar.gz.
    reason: RestoreComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-11-11T14:55:32Z"
    reason: RestoreInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-11-11T10:14:39Z"
    reason: RestoreStatusUnknown
    status: "False"
    type: Warning
  fullBackupName: ""
  id: bybyibd99wxtmkmk
  message: ""
```

c. Check the contents of the restore and make sure that it contains the following statements, which indicate a successful operation:

```
. . .
  reason: RestoreComplete
  status: "True"
  type: Ready
```

If the operation did not complete successfully, see [Troubleshooting Developer Portal backup and restore on OpenShift and Cloud Pak for Integration](#) for information on reviewing logs to help resolve the problem.

Troubleshooting Developer Portal backup and restore on OpenShift and Cloud Pak for Integration

Review details on a failed backup or restore operation for Developer Portal backup on OpenShift and Cloud Pak for Integration.

Check the status of a backup operation

1. Run the following command to get a list of backups:

```
oc get portalbackup -n <Portal_namespace>
```

The response looks like the following example, where only `crType: record` backups include an ID.

NAME	ID	STATUS	TYPE	CR TYPE	AGE	COMMENT
pl-bup-gqz8f	20200526.142449	Ready	system	record	55m	

p1-bup-jx4vc	20200526.142458	Ready	site	record	54m	
p1-bup-st4l2		Ready	all	create	56m	y
p1-bup-th2dm		Ready	all	create	4h51m	y

2. Run the following command to view the details of a particular backup,, which is specified using the <bup-name> from the **NAME** column in the list of backups:

```
oc get portalbackup <bup-name> -n <Portal_namespace> -o yaml
```

The details of a backup look like the following example:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalBackup
metadata:
  creationTimestamp: "2020-10-01T10:00:16Z"
  generateName: portal-bup-
  generation: 1
  name: portal-bup-g66gv
  namespace: default
  resourceVersion: "8940"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/default/portalbackups/portal-bup-g66gv
  uid: b1d5b295-b966-48f2-b6ef-dfcd1c499741
spec:
  comment: test comment
  portalCluster: portal
  crType: create
  siteName: installed
  type: site
status:
  backupId: ""
  commentLeft: "y"
  conditions:
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    message: |-
      2020-10-01 10:00:18: CLI task (backup:site) starting.
      2020-10-01 10:00:20: Making a backup for site conor.com
      2020-10-01 10:00:27: A local backup was successfully created: /var/aegir/backups/conor.com-20201001.100021.tar.gz
      2020-10-01 10:00:30: Attempting to upload backup to remote backup server
      2020-10-01 10:00:30: Uploading conor.com-20201001.100021.tar.gz to s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-
      standard:22:cq-test/ptlcq4/ using objstore
      2020-10-01 10:00:33: Site with URL conor.com (conor.com-20201001.100021.tar.gz) successfully uploaded
      2020-10-01 10:00:33: All installed sites have been successfully backed up
      2020-10-01 10:00:33: CLI task (backup:site) completed successfully.
    reason: BackupComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    reason: BackupInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupStatusUnknown
    status: "False"
    type: Warning
  fullBackupName: ""
  id: f8t0ux7z4zfnjnju
  message: ""
  phase: Ready
  state: ""
```

3. If the operation failed, check the associated **message** for more information.

Check the status of a restore operation

1. Run the following command to get the name of the most recent restore record:

```
oc get portalrestore -n <Portal_namespace>
```

The response looks like the following example:

NAME	READY	AGE
portal-restore-twjrjrp	Ready	28s

2. Run the following command to view information about the restore operation; which is indicated with the <portal_restore_name> shown in the **NAME** column:

```
oc get portalrestore <portal_restore_name> -n <Portal_namespace> -o yaml
```

The details of a failed restore operation look like the following example:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalRestore
metadata:
  creationTimestamp: "2020-05-26T13:10:46Z"
  generateName: portal-restore-
  generation: 1
  name: portal-restore-x2bf9
  namespace: portal
  resourceVersion: "1493471"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/portal/portalrestores/portal-restore-x2bf9
  uid: 9f6a1e22-55ab-4595-8b98-bdfacec40d58c
```

```

spec:
  type: site
  portalCluster: portal
status:
  backupId: ""
  commentLeft: ""
  conditions:
  - lastTransitionTime: "2020-05-26T13:10:49Z"
    message: |
      Please provide a site to restore. To restore all sites use value 'all' in the CR
    status: Failed
    type: Ready
  fullBackupName: ""
  id: ""
  message: ""

```

3. If the operation failed, check the associated **message** for more information.

Review the logs for additional information

Look for information in the logs of the `ibm-apiconnect` pod:

- Locate the `ibm-apiconnect` pod with:

```
oc get pods -n <Portal_namespace>
```

- View the logs with:

```
oc logs <ibm-apiconnect-pod-NAME> -n <Portal_namespace>
```

Backing up and restoring the Analytics database on OpenShift and Cloud Pak for Integration

The IBM® API Connect Analytics database can be backed up and restored from an S3 repository. S3 compatible object storage is required; for example, IBM Cloud Object Storage.

About this task

Before you can initiate a backup or restore operation, you must update the Analytics CR (custom resource) to configure your settings. Generating a backup for Analytics is performed on demand by deploying a backup CR. You can deploy a restore CR to start the restoration of a previously generated backup.

Note: On a v10.0.5 API Connect deployment, you can restore only analytics backups that were taken from a v10.0.5 analytics deployment. You cannot restore analytics backups that were taken from a deployment that was on v10.0.4 or an earlier release.

- [Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration](#)
Configure the IBM API Connect Analytics subsystem on OpenShift and Cloud Pak for Integration to support backing up and restoring the Analytics database.
- [Running a backup of the Analytics database on OpenShift and Cloud Pak for Integration](#)
Deploy the `AnalyticsBackup` CR to initiate a backup of the IBM API Connect Analytics database on OpenShift or Cloud Pak for Integration.
- [Restoring the Analytics database on OpenShift and Cloud Pak for Integration](#)
Deploy the `AnalyticsRestore` CR to initiate restoration of a backup of the IBM API Connect Analytics database on OpenShift or Cloud Pak for Integration.

Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration

Configure the IBM® API Connect Analytics subsystem on OpenShift and Cloud Pak for Integration to support backing up and restoring the Analytics database.

About this task

Generating a backup for Analytics is performed on demand by configuring backup settings in the Analytics section of the top-level `APIConnectCluster` CR. You can create a restore CR to start the restoration of a previously generated backup.

- Backups can be scheduled or generated manually (on-demand), but you can only run one backup or restore process at a time.
- The backup and restore solution for Analytics supports only remote, cloud-based S3 storage (such as IBM Cloud Object Storage, or AWS S3).

Note:

- Only host-style S3 backups are supported. Path-style S3 backups are not supported.
- If you are using a self-managed S3 backup server, check that you have a DNS hostname entry and matching TLS certificate for your S3 backup endpoint.

Procedure

1. Create the analytics backup secret for your S3-compatible storage:

- a. Obtain an access key and corresponding access key secret from your S3 provider.

For example, you might obtain the key and secret from one of the following providers, using the instructions found on their sites:

- [IBM Cloud Object Storage](#)
- [AWS](#)

b. Create the Kubernetes secret by running the following command and filling in your access key, access key secret, and namespace:

```
oc create secret generic analytics-backup-secret --from-literal=access_key='Your_Access_Key' --from-literal=secret_key='Your_access_key_secret' -n Analytics_namespace
```

2. Add the backup configuration to the Analytics section of the top-level `APIConnectCluster` CR:

You can add or modify the Analytics backup configuration using one of the following methods:

- OpenShift web console:
 - Navigate to Advanced Configuration, > Analytics Subsystem, > Advanced Configuration, > Database Backups, > Advanced Configuration.
 - Configure the backup settings using the settings described in Table 1.
- IBM Cloud Pak Platform UI:
 - Open the API Connect instance for editing.
 - On the Details page, enable Advanced settings, which allows you to add backup configurations in the Analytics subsystem section.
 - Click Analytics subsystem and configure settings in the "Database backups" section using the settings described in Table 1.
- YAML: Edit the top-level `APIConnectCluster` CR and add a `databaseBackup` section using the settings described in Table 1; for example:

```
spec:
  analytics:
    databaseBackup:
      credentials: analytics-backup-secret
      host: s3.eu-gb.cloud-object-storage.appdomain.cloud
      path: my-s3-bucket/apic-analytics-backups
      backupCerts: <custom-s3-server-CA-cert>
      schedule: "0 2 * * *"
      enableCompression: true
      chunkSize: 1GB
      enableServerSideEncryption: false
```

Table 1. Backup configuration settings for Analytics

Field name in UI	Field name in YAML	Description
Credentials	<code>credentials</code>	The name of the Kubernetes secret that you created in step 1.
Server hostname	<code>host</code>	The S3 endpoint with the corresponding S3 region in the format <code>s3.s3region.s3domain</code>
Backup path	<code>path</code>	A combination of the S3 bucket and the base path within the bucket, using the format <code>bucket_name/base_path</code>
Schedule	<code>schedule</code>	(Optional) The schedule that determines how often backups will be invoked automatically. The format for the schedule is any valid cron string. The timezone for backups is that of the kube-controller-manager. Backing up once a day is probably sufficient; remember that the more frequently you run backups, the more storage space you need.
Compress	<code>enableCompression</code>	(Optional) Determines whether metadata files are stored in compressed format. Default value is <code>true</code> .
Backup chunk size	<code>chunkSize</code>	(Optional) Specifies how large files are stored. Large files can be stored as chunks when the snapshot is created. This setting specifies the size of the chunks as GB, MB, or KB. Default value is <code>1GB</code> .
Enable server side encryption	<code>enableServerSideEncryption</code>	(Optional) Determines whether files are encrypted. When set to true, files are encrypted on the server side using AES256. Default value is <code>false</code> .
Backup Certs	<code>backupCerts</code>	<p>The name of a custom certificate that contains your upstream custom S3 CA certificate. Supported beginning with 10.0.2. This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be <code>ca.crt</code> and the value is the base64-encoded value of the CA certificate.</p> <p>If the server certificate that is presented by your S3/object-store endpoint is signed by a well known CA and includes any intermediate certificates in the chain, then you do not need to provide the CA certificate using this feature because the Analytics subsystem already trusts the server certificate. You can test this by configuring the Analytics backup to your S3/object-store server without providing the CA certificate.</p> <p>If you do need to provide the CA certificate then you must provide the entire chain in the <code>ca.crt</code> member of the secret, as follows:</p> <pre>intermediate-certificate-1 intermediate-certificate-2 . . . intermediate-certificate-n root-certificate</pre> <p>Where the first certificate in the <code>ca.crt</code> member (intermediate-certificate-1) is the issuer of the S3/object-store server certificate, and each subsequent certificate in the <code>ca.crt</code> member is the issuer of the preceding certificate. The root certificate in the <code>ca.crt</code> member must be self-signed.</p> <p>If there are no intermediate certificates involved, then the <code>ca.crt</code> member contains only the root certificate.</p> <p>The following sample bash script creates the Kubernetes secret:</p> <pre>cat >customs3ca.yaml <<EOF apiVersion: v1 data: ca.crt: \$(base64 <path-to-ca-certificate> tr -d '\n') kind: Secret metadata: name: custom-server-ca type: generic EOF kubect1 apply -f customs3ca.yaml -n <namespace></pre>

Running a backup of the Analytics database on OpenShift and Cloud Pak for Integration

Deploy the **AnalyticsBackup** CR to initiate a backup of the IBM® API Connect Analytics database on OpenShift or Cloud Pak for Integration.

Before you begin

[Configure backup settings for Analytics.](#)

About this task

A backup is triggered by creating the **AnalyticsBackup** CR. The new CR is detected by the **ibm-apiconnect** operator, which requests a new backup.

Procedure

1. Make sure your analytics cluster status is **Running** and **READY**.

The cluster is ready when the **READY** condition states **x/y** where **x = y**. You can check the status in the OpenShift Form UI, or by running the following command:

```
oc get a7s -n <analytics namespace>
```

where **a7s** is the name of your Analytics subsystem. The response looks like the following example:

NAME	READY	STATUS	VERSION	RECONCILED VERSION	AGE
analytics	5/5	Running	10.0.5.0	10.0.5.0-426	19h

2. Create the **AnalyticsBackup** CR using either the IBM Cloud Pak Platform UI, the OpenShift web console, or the CLI.

For example:

```
apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsBackup
metadata:
  generateName: a7s-bup-
  namespace: <analytics namespace>
spec:
  crType: create
  comment: "test comment"
  indices:
  - all
  enableIgnoreUnavailable: true
```

where:

- **metadata.generateName** - Use to specify a prefix for your backup name, the remainder of the name will be generated at runtime.
- **metadata.name** - Use as an alternative to **generateName**, where you want to specify the full name of the backup.
- Optional: **metadata.namespace** - Specify the analytics namespace.
- **spec.crType** - Use **create** to trigger a backup process. When the backup is created, the operator generates another backup CR with **crType: record**, which represents the stored backup that you can use for a restore operation.
- Optional: **spec.comment** - You can use this field to describe the back up for later reference.
- Optional: **spec.indices** - A list of index names or keywords, indicating which indices you want to back up. Options are: **all | apievents | ui**, or the names of individual indices. If no indices are specified, then all indices are backed up. The default of 'all' is specified explicitly in the above sample.
- Optional: **spec.enableIgnoreUnavailable** - Default value is **false**. If set to **false**, the backup fails if a specified index is not present. If set to **true**, missing indices are skipped and the backup continues.

3. Trigger the backup operation by clicking Create in the UI, or by running the following command to deploy the CR in the namespace for your Analytics subsystem:

```
oc create -f analyticsbackup_cr.yaml -n <Analytics-namespace>
```

The backup's name and ID are generated when the backup operation runs.

4. To get a list of available backups, run the following command:

```
oc get analyticsbackup -n <Analytics-namespace>
```

where **<Analytics-namespace>** is the namespace where the Analytics subsystem is deployed. The response looks like the following example:

NAME	STATUS	ID	CR TYPE	INDICES	AGE	COMMENT
a7s-bup-266p8	Ready		create	[all]	20h	y
a7s-bup-pffdh	Ready		create	[all]	34h	y

5. Display the details for a particular backup by running the following command:

Use the backup name from the results in the previous step.

```
oc get analyticsbackup backup_NAME -n <Analytics-namespace> -o yaml
```

The result looks like the following example:

```
apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsBackup
metadata:
  creationTimestamp: "2022-06-29T12:48:56Z"
  generateName: analytics-bup-
  generation: 1
  name: analytics-bup-4m7g8
```

```

namespace: apic
resourceVersion: "11207730"
uid: e1d1c259-850f-4b90-82b3-6a13c05e5d72
spec:
  comment: ""
  crType: create
  enableIgnoreUnavailable: false
  indices:
  - all
status:
  backupID: analytics-all-2022-06-29t12:48:49utc
  commentLeft: ""
  conditions:
  - lastTransitionTime: "2022-06-30T16:18:24Z"
    message: SUCCESS
    status: Complete
    type: Ready
  details: |
    {Name:my-backup-id
    Version:5.6.16
    Indices:[...]
    StartTime:2022-06-30T16:18:09.698Z
    EndTime:2022-06-30T16:18:16.082Z
    State:SUCCESS
    Failures:[]
    Shards:{Failed:0
    Successful:1}}
  id: my-backup-id

```

Restoring the Analytics database on OpenShift and Cloud Pak for Integration

Deploy the **AnalyticsRestore** CR to initiate restoration of a backup of the IBM® API Connect Analytics database on OpenShift or Cloud Pak for Integration.

Before you begin

[Create a backup](#). If you intend to restore from a recent backup, ensure that the backup is finished before you start the restore operation.

About this task

A restore operation is triggered by creating the **AnalyticsRestore** CR. The new CR is detected by the **ibm-apiconnect** operator, which requests a restore of the specified backup.

Note: You can only run one backup or restore process at a time.

Procedure

- Find the backup that you want to restore.
 - Get a list of available backups by running the following command:

```
oc get analyticsbackup -n <APIC_namespace>
```

where **<APIC_namespace>** is the namespace where the Analytics subsystem is deployed. The response looks like the following example. Note down the ID of the backup that you want to restore.

NAME	STATUS	ID	CR TYPE	INDICES	AGE	COMMENT
a7s-bup-266p8	Ready		create	[all]	20h	y
a7s-bup-pffdh	Ready		create	[all]	34h	y
analytics-bup-kxwq5	Ready	analytics-all-2020-09-07t03:49:15utc	record	[all]	34h	
analytics-bup-p29z8	Ready	analytics-all-2020-09-07t18:04:03utc	record	[all]	20h	

Note: You can only restore the Analytics subsystem from backups where the **CR TYPE** is **record**.

- Create the **AnalyticsRestore** CR using either the IBM Cloud Pak Platform UI, the OpenShift web console, or the CLI. In the CR, specify the ID of the backup that you want to restore. For example:

```

apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsRestore
metadata:
  generateName: a7s-restore-
spec:
  indices:
  - all
  backupID: analytics-all-2022-07-01t03:49:15utc
  enableIgnoreUnavailable: true
  enableOverride: false

```

where:

- indices** - A list of index names or keywords, indicating which indices you want to restore. Options are: **all** | **apievents** | **ui**, or the names of individual indices. If no indices are specified, then all indices are restored. The default of 'all' is specified explicitly in the above sample.
- backupID** - The ID of the backup to be restored. In the example, the backup ID is **analytics-all-2022-07-01t03:49:15utc**.
- enableIgnoreUnavailable** - Default value is **false**. If set to **false**, the restore fails if a specified index is not present in the backup identified by the **backupID**. If set to **true**, missing indices are skipped and the restore continues.
- enableOverride** - Default value is **false**. If set to **true**, then the restore overrides existing indices. This might result in a loss of data from the current database.

When `enableOverride` is set to `false` and you specified indices that already exist in the database that you are restoring, then the restore fails. The failure prevents you from accidentally replacing or losing data that exists in the current database.

3. Trigger the restore operation by clicking Create in the UI, or by running the following command to deploy the CR in the namespace for your Analytics subsystem:

```
oc create -f analyticsrestore_cr.yaml -n <APIC_namespace>
```

The restore's name is generated when the restore operation runs.

4. To get a list of restores, run the following command:

```
oc get analyticsrestore -n <APIC_namespace>
```

The response looks like the following example.

NAME	STATUS	ID	AGE
a7s-restore-01	Running	analytics-all-2026-07-01t12:48:49utc	11s

5. Display the details for a particular restore by running the following command:

Use the restore name from the results in the previous step.

```
oc get analyticsrestore <restore_NAME> -n <APIC_namespace> -o yaml
```

The result looks like the following example,:

```
apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsRestore
metadata:
  generateName: a7s-restore-
  selfLink: >-
  /apis/analytics.apiconnect.ibm.com/v1beta1/namespaces/apic/analyticsrestores/a7s-restore-01
  resourceVersion: '11267266'
  name: a7s-restore-01
  uid: e772c6c9-754a-45da-a7c7-507a56cbb94a
  creationTimestamp: '2022-07-01T12:48:50Z'
  generation: 1
  namespace: apic
spec:
  backupID: 'analytics-all-2026-07-01t12:48:49utc'
  enableIgnoreUnavailable: false
  enableOverride: true
  indices:
    - all
status:
  backupID: ''
  commentLeft: ''
  conditions:
  - lastTransitionTime: "2022-07-01T14:08:03Z"
    message: WARNING
    status: Warning
    type: Ready
  details: |
    {NumberOfDataNodes:1
    Status:yellow
    ActivePrimaryShards:23
    ActiveShards:23
    RelocatingShards:0
    InitializingShards:0
    UnassignedShards:40
    PendingTasks:0}
  id: analytics-all-2022-07-01t14:49:15utc
```

When the restore is successfully started, the `status: condition` section displays `Running`. If the restore does not start correctly, the `status: condition` section displays `Failed`.

The restore process is successful when the storage cluster status is green, there are no unassigned shards, and there are no initializing shards. At that time the `status: condition` section displays `Complete`. If `status: Complete` is not achieved after 3 hours, the status changes to `Warning` and the details are no longer updated.

Attention: For deployment profiles with fewer than 3 storage nodes, the restore process is never marked as `Complete`. The storage cluster status is expected to be yellow and to have more than 0 unassigned shards. In this situation, the restore process is likely to be complete when there are 0 initializing shards.

Disaster recovery

Prepare for, or recover from, a disaster affecting an API Connect deployment running on Kubernetes, OpenShift, or Cloud Pak for Integration.

The backups that you create for disaster recovery are for recovery of the API Connect subsystems in the same environment that the backups were taken, or in a different environment that has the same network configuration and API Connect form factor. If you want to move your API Connect deployment to a different environment and change the form factor or modify your API Connect endpoints, then see [Migrating from v10 to v10 on a different form factor](#).

- [Disaster recovery on Kubernetes](#)
Prepare for disaster events and recover API Connect on Kubernetes if a disaster event occurs.
- [Disaster recovery on OpenShift and Cloud Pak for Integration](#)
Prepare for, or recover from, a disaster affecting an API Connect deployment running on OpenShift or Cloud Pak for Integration

Disaster recovery on Kubernetes

Prepare for disaster events and recover API Connect on Kubernetes if a disaster event occurs.

You must complete preparation steps prior to a disaster in order to recover API Connect on Kubernetes.

Note: For information about how to maintain a two data center deployment, including failover and recovery procedures, see [Maintaining a two data center deployment](#).

- [Preparing for a disaster](#)
Prepare your API Connect deployment for disaster recovery in a Kubernetes environment.
- [Recovering from a disaster](#)
Recover the API Connect subsystems on Kubernetes.

Preparing for a disaster

Prepare your API Connect deployment for disaster recovery in a Kubernetes environment.

Begin preparing for disaster recovery by creating a back-up copy of the installation CR for each subsystem with the following commands:

1. `kubectl -n <namespace> get mgmt <mgmt-subsystem-name> -o yaml > mgmt-subsytem-cr.yaml`
2. `kubectl -n <namespace> get a7s <a7s-subsystem-name> -o yaml > a7s-subsytem-cr.yaml`
3. `kubectl -n <namespace> get ptl <ptl-subsystem-name> -o yaml > ptl-subsytem-cr.yaml`

- [Preparing the management subsystem for disaster recovery](#)
You can prepare a Management subsystem for disaster recovery from S3 or SFTP backups by taking specific steps before and after the installation of the subsystem.
- [Preparing the Developer Portal for disaster recovery](#)
You can prepare a Developer Portal subsystem for disaster recovery by taking specific steps before and after the installation of the subsystem.
- [Preparing the Analytics subsystem for disaster recovery](#)
Prepare the API Connect Analytics subsystem for disaster recovery in a Kubernetes environment.

Preparing the management subsystem for disaster recovery

You can prepare a Management subsystem for disaster recovery from S3 or SFTP backups by taking specific steps before and after the installation of the subsystem.

About this task

- This task must be performed before any disaster event occurs, and also prior to the installation of a replacement Management subsystem during the recovery process. Best practice is to complete these steps immediately after initial configuration of a management subsystem during your original v10 deployment.
- Any `local` backups are presumed to have been lost in the disaster scenario and are non-recoverable in this procedure.

Important: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

Procedure

1. Prior to initial installation, configure the Management subsystem for database backups:
See [Configuring backup settings for fresh install of the Management subsystem](#).
2. After installation, back up essential Kubernetes secrets that are used by the Management subsystem.
The Kubernetes secrets are created during installation. You must save these in case a restore is needed after a disaster has occurred. At that time, you will use them in the setup of the restored Management subsystem:

- a. Obtain and save the Management database encryption secret.

Use the following command to find the name of the secret in the status of the Management Subsystem. Replace `<namespace>` with the namespace used for the subsystem installation:

```
kubectl get mgmt -n <namespace> -o yaml | grep encryption
```

The output of this command shows the name of the Kubernetes secret storing the database encryption key, for example:

```
encryptionSecret: management-enc-key
```

In this case, the name of the Management database encryption secret is `management-enc-key`. Use the following command to back up this secret locally:

```
kubectl get secret management-enc-key -n <namespace> -o yaml > management_enc_key.yaml
```

The secret is stored in a file called `management_enc_key.yaml`, located in the present working directory.

- b. Obtain and save the Management client application credential secrets.

- i. Use the following command to find the names of the secrets in the status of the Management subsystem:

```
kubectl get mgmt -n <namespace> -o yaml | grep CredentialSecret
```

The output of this command shows the names of the Kubernetes Secrets used to store the various client application credentials. For example:

```
atmCredentialSecret: management-atm-cred
consumerToolkitCredentialSecret: management-ccli-cred
consumerUICredentialSecret: management-cui-cred
designerCredentialSecret: management-dsgr-cred
juhuCredentialSecret: management-juhu-cred
toolkitCredentialSecret: management-cli-cred
uiCredentialSecret: management-ui-cred
```

Here, for example, the name of the ATM Client Application Credential Secret is `management-atm-cred`.

- ii. Next, back up all of the secrets locally. Use the following command for each listed Credential Secret, replacing `<secret_name>` with the secret name listed each time:

```
kubectl get secret <secret_name> -n <namespace> -o yaml > <secret_name>.yaml
```

For example, use the following command to backup the ATM Client Credential Secret:

```
kubectl get secret management-atm-cred -n <namespace> -o yaml > management-atm-cred.yaml
```

- iii. After you save each of the client application Credential Secrets locally, open each of the YAML files saved for each secret in turn. Remove both the `ownerReferences` subsection and the `selfLink` property. Re-save the file.

For example, the `ownerReferences` and `selfLink` properties to be removed appear in the YAML files similar to the following:

```
ownerReferences:
- apiVersion: management.apiconnect.ibm.com/v1beta1
  blockOwnerDeletion: true
  controller: true
  kind: ManagementCluster
  name: management
  uid: 623e6b20-7eb8-46ce-94ac-6b64cd71afc4
  selflink: /api/v1/namespaces/default/secrets/management-atm-cred
```

- c. Ensure that all of the YAML files that contain the various backed-up Secrets are stored persistently and safely.

Important: If the files are lost, you cannot restore after a disaster event.

- d. Take note of the following values in the Management subsystem CR. You will need them to restore the Management subsystem.

Table 1. Management subsystem CR settings

Setting	Used with Protocol	Example value
<code>name</code>	S3 and SFTP	<code>management</code>
<code>siteName</code>	S3 and SFTP	<code>82b290a2</code>
<code>originalUID</code>	S3 and SFTP	<code>fa0f6f49-b931-4472-b84d-0922a9a92dfd</code> : During restore, if you do not specify <code>originalUID</code> , the operator automatically sets its value in the new CR to the value of <code>metadata.uid</code> . If the <code>originalUID</code> in the new CR does not match <code>originalUID</code> in the saved CR, the restore will fail.
<code>databaseBackup.protocol</code>	S3 and SFTP only	<ul style="list-style-type: none"> • S3: <code>objstore</code> • SFTP: <code>sftp</code>
<code>databaseBackup.s3provider</code>	S3 only	<code>aws</code>
<code>databaseBackup.host</code>	S3 and SFTP	<ul style="list-style-type: none"> • SFTP <code><SFTP-host-name></code> • S3: <code>s3.amazonaws.com/eu-west-1</code>
<code>databaseBackup.path</code>	S3 and SFTP	<ul style="list-style-type: none"> • SFTP <code><SFTP-path></code> • S3: <code>my-bucket/mgmt</code>
<code>databaseBackup.schedule</code>	S3 and SFTP	<code>0 3 * * *</code>
<code>databaseBackup.credentials</code>	SFTP only	<code>secret-containing-SFTP-credentials</code>
<code>databaseBackup.backupCerts</code>	S3 (custom only, v10.0.2.0 or later)	<code>my-custom-s3-ca-cert</code>
<code>databaseBackup.backups3URISTyle</code>	S3 (custom only, v10.0.2.0 or later)	<code>my-s3-uri-style</code>

- The values `name` and `siteName` form the name of the database cluster name, such as `management-82b290a2-postgres` and is needed to synchronize the backups from your old Management subsystem to your new one. The `siteName` is either a randomly generated alphanumeric ID, or a custom name that was specified by the administrator in the Management Subsystem CR helper YAML file at deploy time. Use the following command to get the `siteName`:

```
kubectl get mgmt -o yaml -n <namespace> | grep siteName
```

- The `databaseBackup.host` and `databaseBackup.path` settings must be identical to what was configured on your old Management subsystem. This is the location of your old Management subsystem backups from where we will recover from.
- Take note of `databaseBackup.schedule`. The schedule will be included during the restoration.

3. Make sure that you have a backup that can be used in case of a disaster event:

- If scheduled backups were configured as part of [Configuring backup settings for fresh install of the Management subsystem](#), backups to the specified cloud storage will take place as scheduled.
- You can also perform a manual backup at any time by creating a manual backup CR. See [Generating a manual backup of the management subsystem](#).

What to do next

You should now complete the preparation steps for the Developer Portal subsystem; see [Preparing the Developer Portal for disaster recovery](#).

Preparing the Developer Portal for disaster recovery

You can prepare a Developer Portal subsystem for disaster recovery by taking specific steps before and after the installation of the subsystem.

About this task

- This task must be performed before any disaster event occurs, and also prior to the installation of a replacement Developer Portal subsystem for the recovery process. Best practice is to complete these steps immediately after initial configuration of a Developer Portal subsystem during your original v10 deployment.
- Any `local` backups are presumed to have been lost in the disaster scenario and are non-recoverable in this procedure.

Important: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. For information about preparing the Management subsystem, see [Preparing the management subsystem for disaster recovery on Kubernetes](#). If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

Procedure

1. Before the initial installation, configure the Developer Portal subsystem for database backups:
 - a. Ensure that the Developer Portal backup subsystem secret is created in the cluster.
This secret contains the credentials (username and password) for the remote storage, such as S3 or IBM Cloud Object Storage, that is used for storing the database backups.
 - b. Ensure that the Developer Portal subsystem is configured for database backups.
Add a `databaseBackups` subsection to the `spec` subsection of the Developer Portal subsystem Custom Resource (CR) YAML file. For example:

```
portalBackup:
  credentials: portal-backup-secret
  host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
  path: test-bucket/restore-test
  port: 443
  protocol: objstore
  schedule: ''
```

See also: [Backing up and restoring the Developer Portal in a Kubernetes environment](#)

2. After installation, you must back up essential Kubernetes secrets that are used by the Developer Portal subsystem.
The Kubernetes secrets are created during installation. You must save these secrets in case a restore is needed after a disaster. Then, you use them in the setup of the restored Developer Portal subsystem:
 - a. Obtain and save the Developer Portal database encryption secret.
Use the following command to find the name of the secret in the status of the Developer Portal Subsystem. Replace `<namespace>` with the namespace used for the subsystem installation:

```
kubectl get portal -n <namespace> -o yaml | grep encryption
```


The output of this command shows the name of the Kubernetes secret that is storing the database encryption key, for example:

```
encryptionSecret: portal-enc-key
```


In this case, the name of the Developer Portal database encryption secret is `portal-enc-key`. Use the following command to back up this secret locally:

```
kubectl get secret portal-enc-key -n <namespace> -o yaml --export > portal_enc_key.yaml
```


The secret is stored in a file that is called `portal_enc_key.yaml`, located in the present working directory.
 - b. Obtain the endpoint definitions as they need to stay the same during the restore.
Get the `portalAdminEndpoint` name:

```
kubectl get portalcluster.portal.apiconnect.ibm.com portal -o jsonpath="{..portalAdminEndpoint.hosts[0].name}"
```


Get the `portalUIEndpoint` name:

```
kubectl get portalcluster.portal.apiconnect.ibm.com portal -o jsonpath="{..portalUIEndpoint.hosts[0].name}"
```


Note: If you have more than one host specified in the `portalUIEndpoint` definition, you must get the name for each one.
 - c. Ensure that all of these YAML files that contain the various backed-up Secrets are stored persistently and safely.
Important: If the files are lost, you cannot restore after a disaster event.
3. Take note of the value of `originalUID` in the Portal subsystem CR. You will need it to restore the Portal subsystem. For example:

```
spec:
  originalUID: "447ea4b3-9514-4a84-a34b-ce0b349838da"
```

Note:

- **Version 10.0.3.0 or later:** During restore, if you do not specify `originalUID`, the operator automatically sets its value in the new CR to the value of `metadata.uid`. If the `originalUID` in the new CR does not match `originalUID` in the saved CR, the restore will fail.
 - **Version 10.0.2 backups:** If planning to restore a Version 10.0.2 backup on Version 10.0.3.0 or later, make a note of the subsystem CR `metadata.uid`. You must use it when performing a restore on v10.0.3.0 or later, to populate the `spec.originalUID` value of the CR.
 - **Limitation for Version 10.0.2:** If planning to restore on Version 10.0.2, performing a restore may not work if the subsystem CR name exceeds 15 characters.
4. If Scheduled backups are set as part of the `databaseBackup` subsection in the Developer Portal Subsystem CR helper YAML file (`portal_cr.yaml`), backups to the specified cloud storage take place.
You can also complete a manual backup at any time by creating a manual backup CR.

a. To complete a manual backup, use the Developer Portal backup CR helper YAML file (`portalbackup_cr.yaml`). For example:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalBackup
metadata:
  generateName: portal-bup-
spec:
  type: all
  crType: create
  comment: "test comment"
```

Run the following command to start a manual backup to the configured cloud storage:

```
kubectl create -f portalbackup_cr.yaml -n <namespace>
```

b. Use the following command to list the backups:

```
kubectl get portalbackup -n <namespace>
```

For example:

```
loginabc@localhost ibm-apiconnect]$ kubectl get pb
NAME                ID                STATUS    TYPE    CR TYPE    AGE    COMMENT
portal-bup-g66gv    20201001.100021  Ready    site    create    5m27s  y
portal-bup-z7wgk    20201001.100021  Ready    site    record    4m24s
```

With stored backups, secrets and `siteName` saved, the stack is now ready for a restore after a disaster event.

What to do next

To restore the Developer Portal on Kubernetes after a disaster, see [Recovering the Developer Portal after a disaster](#).

Preparing the Analytics subsystem for disaster recovery

Prepare the API Connect Analytics subsystem for disaster recovery in a Kubernetes environment.

About this task

Preparing for disaster recovery involves backing up data and making a local copy of the Custom Resource (CR) that was used for installing the Analytics subsystem.

Procedure

1. [Perform a backup of the Analytics database](#).

Scheduling automatic backups ensures that a recovery includes recent updates. For more information on configuring backups, see [Configuring backup settings for Analytics](#).

2. Make a copy of the AnalyticsCluster CR (custom resource).

a. Find the Analytics cluster's name by running the following command and looking for the value in the **NAME** column:

```
kubectl get AnalyticsCluster
```

The response looks like the following example:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
gj01-analytics	6/6	Running	10.0.1-eus	10.0.1.1-1402-eus		22h

In this example the cluster's name is `gj01-analytics`.

- b. Get the YAML that describes the Analytics cluster, and save it in a file using the following command:

Use the name of the AnalyticsCluster that you obtained in the previous step.

```
kubectl get AnalyticsCluster gj01-analytics -o yaml > gj01-analytics.yaml
```

- c. Save the file locally.

Recovering from a disaster

Recover the API Connect subsystems on Kubernetes.

- [Recovering the management subsystem on Kubernetes](#)
Recover the management subsystem on Kubernetes.
- [Recovering the Developer Portal after a disaster](#)
You can restore the Developer Portal subsystem after a disaster event.
- [Recovering the Analytics subsystem after a disaster](#)
Recover the API Connect Analytics subsystem in a Kubernetes environment.

Recovering the management subsystem on Kubernetes

Recover the management subsystem on Kubernetes.

The recovery procedures for the management subsystem are specific to the type of backup. Use the instructions for your backup type:

- [Recovering the management subsystem from S3 backups](#)
You can recover the management subsystem from S3 backups after a disaster event.
- [Recovering the management subsystem from SFTP backups](#)
You can recover the management subsystem from SFTP backups after a disaster event.

Recovering the management subsystem from S3 backups

You can recover the management subsystem from S3 backups after a disaster event.

Before you begin

To successfully recover the management subsystem, you must have previously completed the steps in [Preparing the management subsystem for disaster recovery](#). Important: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

About this task

To recover from a disaster event, you must create a new IBM API Connect® installation with a running IBM API Connect Operator.

Note: Limitation for backups created on Version 10.0.2

- If restoring a Version 10.0.2 backup onto a new Version 10.0.2 deployment, performing a restore may not work if the subsystem CR name exceeds 15 characters. This limitation applies only to restoring onto Version 10.0.2.
- Restoration is supported for Version 10.0.2 backups onto a Version 10.0.3.0 or later deployment when subsystem name exceeds 15 characters, as long as the correct `spec.originalUID` is specified upon restore. See Step [3.f](#).

Procedure

1. Determine which backup to restore from.

- For IBM®'s Cloud Object Storage, you can check currently stored backups in the COS console. Select Buckets > Objects. See the Object Names displayed on the `<cos_name>/backup/db` panel. For example:

```
20200605-105429F
20200605-100008F
20200605-11040F
```

- For Amazon's AWS, you can check currently stored backups in the S3 console. For example, under Amazon S3 > `cluster_name` > old cluster > backup > db:

```
20200606-144315F
20200606-145011F
```

The format of the backup ID is `YYYYMMDD-HHMMSS<F|I>`

Take note of the ID of the backup you wish to restore to. Each ID contains the date, time, and type of the backup stored. This will be used later in the procedure.

- Incremental backups are denoted with a suffix `I` on the ID, ensure each incremental backup has its prior backup also present in storage. You can check this by examining the ID `<prior-backup-id>_<backup-id>`.
- Full backups are denoted with a suffix `F` on the ID.

Note:

During the disaster recovery process, the S3 configuration detail of the older management system is used, but the older management system must be in offline mode. The old subsystem must be offline because you cannot have two management systems simultaneously using the same S3 bucket name in the database backup configurations.

2. Make sure you know the management database cluster name.

You can get this name from the original management subsystem CR. You made note of this name in Step [2.d](#) in [Preparing the management subsystem for disaster recovery](#).

If you are not able to recover the original management subsystem CR, use the following steps to recover the Management database cluster name:

- Open your IBM Cloud® Object Storage or AWS S3 console and proceed to the bucket location where the old Management subsystem backups are located.
- Download `backup/db/<backup-id>/pg_data/postgresql.conf.gz`. Open `postgresql.conf` to view the database cluster name:
 - IBM Cloud Object Storage

```
.
.
cluster_name = 'm1-a6287572-postgres'
.
```

- `m1` - Management subsystem name
 - `a6287572` - site name
- AWS S3

```
.
.
```

```
cluster_name = 'm1-c91dc0b9-postgres'
```

- `m1` - Management subsystem name
- `c91dc0b9` - site name

3. Before installing the replacement management subsystem CR:

- a. Apply the YAML file that contains the Management Database Encryption Secret into the cluster. For example, where `encryption-bin-secret.yaml` is the local YAML file containing the backup-up encryption secret:

```
kubectl apply -f encryption-bin-secret.yaml -n <namespace>
```

Replace `<namespace>` with the namespace being used for the management subsystem installation.

This command re-creates the original Management Database encryption secret on the cluster. It will be named as the original name of the secret.

- b. Add the following `encryptionSecret` subsection to the `spec` of the Management CR. For example, if `management-enc-key` is the name of the newly created secret on the cluster containing the original Management Database encryption secret from the previous step:

```
encryptionSecret:
  secretName: management-enc-key
```

- c. For each of the saved YAML Files that contain the Management Client Application Credential Secrets, apply each file into the cluster using the following command:

```
kubectl create -f <secret_name>.yaml -n <namespace>
```

where `<secret_name>` is the local YAML file containing one of the backed-up Credential Secrets.

Repeat this for each of the backed-up Credential Secrets. These are the secrets you saved in Step 2.b in [Preparing the management subsystem for disaster recovery](#).

These commands will re-create the original Management Client Application Credential Secrets on the cluster. Each will be named as the original name of the Secret.

- d. Add the following `customApplicationCredentials` subsection to the `spec` subsection of the Management CR:

```
customApplicationCredentials:
- name: atm-cred
  secretName: management-atm-cred
- name: ccli-cred
  secretName: management-ccli-cred
- name: cli-cred
  secretName: management-cli-cred
- name: cui-cred
  secretName: management-cui-cred
- name: dsgr-cred
  secretName: management-dsgr-cred
- name: juhu-cred
  secretName: management-juhu-cred
- name: ui-cred
  secretName: management-ui-cred
```

For each named credential above, the `secretName` is given as the corresponding name of the newly created secret from Step 3.c.

- e. Add the `siteName` property to the `spec` of the Management CR.

For example, if `a2a5e6e2` is the original `siteName` that was noted after the installation of the original Management Subsystem:

```
siteName: a2a5e6e2
```

- f. **Version 10.0.3.0 or later:** Add the `originalUID` property to the `spec` of the Management CR.

When recreating a system, to restore a backup into it, you must specify the same `spec.originalUID` in the CR as was present in the system that was backed up. If the `spec.originalUID` in the new CR for recreating the system does not match the `spec.originalUID` that was present in the system that was backed up, the restore will fail.

```
spec:
  originalUID: "fa0f6f49-b931-4472-b84d-0922a9a92dfd"
```

Note:

- For Version 10.0.3.0 or later, if you do not specify `spec.originalUID` in the new CR, the operator automatically sets the Management CR value of `spec.originalUID` to match the new CR value `metadata.uid`. In this case, the restore will fail because the `spec.originalUID` in the saved (backed-up) CR does not match `spec.originalUID` in the new CR.
- The originalUID is only essential when the subsystem CR name exceeds 15 characters in length, or 10 characters limit for the API Connect Cluster CR. Recommended practice is that all backups should include the originalUID for Management.
- See also Step 2.d in [Preparing the management subsystem for disaster recovery](#).

- g. Verify that the name of the management subsystem in the CR matches with the old management subsystem name, as described in Step 2.d in [Preparing the management subsystem for disaster recovery](#).

4. Install the Management subsystem CR with the values obtained in Step 2.d in [Preparing the management subsystem for disaster recovery](#).

Important: The hostnames of the endpoints cannot be changed, and must remain the same in the Management CR YAML file used for installation now as they were for the original installation.

To review installation of the management subsystem, see [Installing the Management subsystem cluster](#).

Once the Management subsystem is installed you may notice `backup` job pods and `stanza-create` job pods in `Error` state.

<code>m1-82b290a2-postgres-stanza-create-4zcgz</code>	0/1	Error	0	35m
<code>m1-82b290a2-postgres-full1-sch-backup-2g9hm</code>	0/1	Error	0	20m

This is expected behavior.

- The `stanza-create` job normally expects buckets or subdirectories within buckets to be empty. However, since we have configured the Management subsystem with a pre-populated bucket (ie. where our backups exist), the job will go into **Error** state.
- Any scheduled or manual backups will go into **Error** state. While we have configured the Management subsystem with our already populated S3 bucket, the new database isn't yet configured to write backups into remote storage.

Important:

For S3, the recovery remains in an intermediate state until the restore is complete, and Postgres wal files might cause serious disk issues. To avoid this possibility, continue immediately with the next step.

Note that if you delay completion of the restore:

- Health check might fail. In this case, you can still proceed to the next step and perform a restore.
 - Postgres wal files might cause problems by consuming all disk space. In this case, you must either:
 - Re-install the system, prepare again for disaster recovery, and perform the restore.
 - Or increase disk space so that the system returns to a stable state, and then proceed with the restore.
5. Get a list of available backups and confirm that the backup ID noted in Step 4 is in the backup list.

The API Connect Operator automatically reads backups from configured remote storage and populates the list of available backups we can restore to.

```
$ kubectl get mgmtb
NAME                STATUS      ID                CLUSTER          SUBSYSTEM  TYPE  CR TYPE  AGE
mgmt-backup-4z87f  Complete   20200606-145011F  m1-82b290a2-postgres  m1        full  record  11m
mgmt-backup-6bms2  Complete   20200606-144315F  m1-82b290a2-postgres  m1        full  record  11m
```

6. Update the Management subsystem CR `databaseBackup.schedule` to what was noted in Step 2.d in [Preparing the management subsystem for disaster recovery](#).
7. Perform a Management Restore using the name of the backup that has the ID you want to restore. For example, as shown in Step 5, for ID `20200606-145011F` the backup name is `mgmt-backup-4z87f`.
For more info on restoring the management subsystem, see [Restoring the management subsystem \(v10.0.1.1 or later\)](#).
8. Use the following command to check the status of the restore:

```
kubectl get mgmtr -n <namespace>
```

Once the Management Restore has completed and the database is running again, the data of the old Management subsystem will be successfully restored onto the new Management subsystem. Note:

- Manual and scheduled backups should perform as normal once again
 - `stanza-create` jobs will continue to report **Error** state as stated in Step 4.
9. Verify that the Cloud Manager UI can now be logged-in to as before, and that Provider Orgs exist as before.
The restore from the disaster event is now complete.

What to do next

You should now complete the recovery steps for the Developer Portal subsystem on Kubernetes, see [Recovering the Developer Portal after a disaster](#).

Recovering the management subsystem from SFTP backups

You can recover the management subsystem from SFTP backups after a disaster event.

Before you begin

To successfully recover the management subsystem, you must have previously completed the steps in [Preparing the management subsystem for disaster recovery](#).
Important: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

About this task

To recover from a disaster event, you must create a new IBM API Connect installation with a running IBM API Connect Operator.

Note: Limitation for backups created on Version 10.0.2

- If restoring a Version 10.0.2 backup onto a new Version 10.0.2 deployment, performing a restore may not work if the subsystem CR name exceeds 15 characters. This limitation applies only to restoring onto Version 10.0.2.
- Restoration is supported for Version 10.0.2 backups onto a Version 10.0.3.0 or later deployment when subsystem name exceeds 15 characters, as long as the correct `spec.originalUID` is specified upon restore. See Step 3.f.

Procedure

1. Determine which backup to restore from.

View the SFTP backups available on your remote storage site:

```
-rw-r--r-- 1 root root 13092333 Aug 26 08:56 20200826-154646F.tgz
-rw-r--r-- 1 root root 18703758 Aug 26 09:10 20200826-160010F.tgz
-rw-r--r-- 1 root root 24318561 Aug 26 09:21 20200826-161301F.tgz
```

Take note of the backup ID of the backup you wish to restore to. Each filename contains the date, time, and type of the backup stored. This will be used later in the procedure.

The format of the backup ID is **YYYYMMDD-HHMMSS<F|I>**. For example, if we want to use the Aug 26, 09:21 backup, its backup ID will be **20200826-161301F**

- a. Incremental backups are denoted with a suffix **I** on the ID, ensure each incremental backup has its prior backup also present in storage. You can check this by examining the ID **<prior-backup-id>_<backup-id>**.
 - b. Full backups are denoted with a suffix **F** on the ID.
2. Make sure you know the management database cluster name.
You can get this name from the original management subsystem CR. You made note of this name in Step [2.d](#) in [Preparing the management subsystem for disaster recovery](#).

If you are not able to recover the original management subsystem CR, you can also recover the Management database cluster name and siteName by examining the SFTP backup tar:

- a. Download or move the SFTP backup tar file and decompress (untar) it.
- b. Open **<management-subsystem-name>-<siteName>-postgres-backrest-shared-repo/backup/db/<backup-id>/pg_data/postgresql.conf.gz** which contains the management subsystem name and siteName. For example:

```
# Do not edit this file manually!
# It will be overwritten by Patroni!
include 'postgresql.base.conf'

archive_command = 'source /opt/cpm/bin/pgbackrest/pgbackrest-set-env.sh && pgbackrest archive-push "%p"'
archive_mode = 'True'
archive_timeout = '60'
autovacuum_vacuum_cost_limit = '1000'
autovacuum_vacuum_scale_factor = '0.01'
cluster_name = 'm1-f785a3e3-postgres'
```

In this example:

- **cluster_name** has both the management subsystem name and siteName
 - **m1** - Management subsystem name
 - **f785a3e3** - site name
3. Before installing the replacement management subsystem CR:
- a. Apply the YAML file that contains the Management Database Encryption Secret into the cluster. For example, where **encryption-bin-secret.yaml** is the local YAML file containing the backup-up encryption secret:

```
kubectl create -f encryption-bin-secret.yaml -n <namespace>
```

Replace **<namespace>** with the namespace being used for the management subsystem installation.

This command re-creates the original Management Database encryption secret on the cluster. It will be named as the original name of the secret.

- b. Add the following **encryptionSecret** subsection to the **spec** of the Management CR. For example, if **management-enc-key** is the name of the newly created secret on the cluster containing the original Management Database encryption secret from the previous step:

```
encryptionSecret:
  secretName: management-enc-key
```

- c. For each of the saved YAML Files that contain the Management Client Application Credential Secrets, apply each file into the cluster using the following command:

```
kubectl create -f <secret_name>.yaml -n <namespace>
```

where **<secret_name>** is the local YAML file containing one of the backed-up Credential Secrets.

Repeat this for each of the backed-up Credential Secrets. These are the secrets you saved in Step [2.b](#) in [Preparing the management subsystem for disaster recovery](#).

These commands will re-create the original Management Client Application Credential Secrets on the cluster. Each will be named as the original name of the Secret.

- d. Add the following **customApplicationCredentials** subsection to the **spec** subsection of the Management CR:

```
customApplicationCredentials:
- name: atm-cred
  secretName: management-atm-cred
- name: ccli-cred
  secretName: management-ccli-cred
- name: cli-cred
  secretName: management-cli-cred
- name: cui-cred
  secretName: management-cui-cred
- name: dsgr-cred
  secretName: management-dsgr-cred
- name: juhu-cred
  secretName: management-juhu-cred
- name: ui-cred
  secretName: management-ui-cred
```

For each named credential above, the **secretName** is given as the corresponding name of the newly created secret from Step [3.c](#).

- e. Add the **siteName** property to the **spec** of the Management CR.

For example, if **a2a5e6e2** is the original **siteName** that was noted after the installation of the original Management Subsystem:

```
siteName: a2a5e6e2
```

- f. **Version 10.0.3.0 or later:** Add the **originalUID** property to the **spec** of the Management CR.

When recreating a system, to restore a backup into it, you must specify the same **spec.originalUID** in the CR as was present in the system that was backed up. If the **spec.originalUID** in the new CR for recreating the system does not match the **spec.originalUID** that was present in the system that was backed up, the restore will fail.

```
spec:
  originalUID: "fa0f6f49-b931-4472-b84d-0922a9a92dfd"
```

Note:

- For Version 10.0.3.0 or later, if you do not specify `spec.originalUID` in the new CR, the operator automatically sets the Management CR value of `spec.originalUID` to match the new CR value `metadata.uid`. In this case, the restore will fail because the `spec.originalUID` in the saved (backed-up) CR does not match `spec.originalUID` in the new CR.
 - The originalUID is only essential when the subsystem CR name exceeds 15 characters in length, or 10 characters limit for the API Connect Cluster CR. Recommended practice is that all backups should include the originalUID for Management.
 - See also Step 2.d in [Preparing the management subsystem for disaster recovery](#).
- g. Verify that the name of the management subsystem in the CR matches with the old management subsystem name, as described in Step 2.d in [Preparing the management subsystem for disaster recovery](#).
4. Install the Management subsystem CR with the values obtained in Step 2.d in [Preparing the management subsystem for disaster recovery](#).
Important: The hostnames of the endpoints cannot be changed, and must remain the same in the Management CR YAML file used for installation now as they were for the original installation.
To review installation of the management subsystem, see [Installing the Management subsystem cluster](#).
5. Get a list of available backups and confirm that the backup ID noted in Step 1 is in the backup list.
The API Connect Operator automatically reads backups from configured remote storage and populates the list of available backups we can restore to.

```
$ kubectl get mgmtb
NAME                STATUS    ID                CLUSTER                SUBSYSTEM    TYPE    CR TYPE    AGE
mgmt-backup-4z87f  Complete  20200606-145011F  m1-82b290a2-postgres  m1           full   record    11m
mgmt-backup-6bms2  Complete  20200606-144315F  m1-82b290a2-postgres  m1           full   record    11m
```

6. Perform a Management Restore using the name of the backup that has the ID you want to restore. For example, as shown in Step 5, for ID `20200606-145011F` the backup name is `mgmt-backup-4z87f`.
For more info on restoring the management subsystem, see [Restoring the management subsystem \(v10.0.1.1 or later\)](#).
7. Use the following command to check the status of the restore:

```
kubectl get mgmtr -n <namespace>
```

Once the Management Restore has completed and the database is running again, the data of the old Management subsystem will be successfully restored onto the new Management subsystem. Manual and scheduled backups should perform as normal once again.

What to do next

You should now complete the recovery steps for the Developer Portal subsystem on Kubernetes, see [Recovering the Developer Portal after a disaster](#).

Recovering the Developer Portal after a disaster

You can restore the Developer Portal subsystem after a disaster event.

Before you begin

Important: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

About this task

To recover from a disaster event, you must create a new IBM API Connect installation with a running IBM API Connect Operator.

Note: Limitation for backups created on Version 10.0.2:

- If restoring a Version 10.0.2 backup onto a new Version 10.0.2 deployment, performing a restore may not work if the subsystem CR name exceeds 15 characters. This limitation applies only to restoring onto Version 10.0.2.
- Restoration is supported for Version 10.0.2 backups onto a Version 10.0.3.0 or later deployment when subsystem name exceeds 15 characters, as long as the correct `spec.originalUID` is specified upon restore. See Step 1.f.

Procedure

1. To prepare for the restoration, you must complete some steps before you reinstall the Developer Portal subsystem.

Warning:

- Ensure that the hostnames of the endpoints remain the same in the Developer Portal subsystem that is used for installation now as they were for the original installation. The hostnames for the endpoints cannot be changed.
- a. Ensure that the Developer Portal backup subsystem secret is created in the cluster.
This secret contains the credentials (username and password) for the remote storage, such as S3 or IBM Cloud Object Storage, that is used for storing the database backups. This secret must be the same secret as used for the original installation.
- b. Ensure that the Developer Portal subsystem is configured for database backups.
Configuration is done by adding a `databaseBackups` subsection to the `spec` subsection of the Developer Portal subsystem Custom Resource (CR) YAML file, `portal_cr.yaml`. The file is located in the `helper_files` archive that was extracted as part of the preparation for installation. See [Backing up and restoring the Developer Portal in a Kubernetes environment](#).

For example:

```
portalBackup:
  credentials: portal-backup-secret
  host: <backup-host>
  path: <backup-path>
  port: 22
  protocol: sftp
  schedule: ''
```

The `host` and `path` must be the same as was specified for the original installation so that the IBM API Connect Operator can access the saved Developer Portal database backups.

- c. Apply the saved YAML file that contains the Developer Portal database encryption secret into the cluster. Use the following command, where `portal_enc_key.yaml` is the local YAML file that contains the backup encryption secret. Replace `<namespace>` with the namespace that is being used for subsystem installation:

```
kubectl apply -f portal_enc_key.yaml -n <namespace>
```

This command re-creates the original Developer Portal database encryption secret on the cluster, which is named as the original name of the secret.

- d. Add the `encryptionSecret` to the Developer Portal subsystem Custom Resource (CR) YAML file, `portal_cr.yaml`. For example, add the following `encryptionSecret` subsection to the `spec` section. In this example, `portal-enc-key` is the name of the new created secret on the cluster that contains the original Developer Portal Database Encryption Secret. The name of the secret that is created in Step [1.a](#) is used here:

```
encryptionSecret:
  secretName: portal-enc-key
```

- e. Ensure the name `spec.metadata.name` of the new Developer Portal cluster CR is the same as the previous Developer Portal CR.
- f. **Version 10.0.3.0 or later:** Add the `originalUID` property to the `spec` of the Portal CR. When recreating a system, to restore a backup into it, you must specify the same `spec.originalUID` in the CR as was present in the system that was backed up. If the `spec.originalUID` in the new CR for recreating the system does not match the `spec.originalUID` that was present in the system that was backed up, the restore will fail.

```
spec:
  originalUID: "447ea4b3-9514-4a84-a34b-ce0b349838da"
```

Note:

- For Version 10.0.3.0 or later, if you do not specify `spec.originalUID` in the new CR, the operator automatically sets the Portal CR value of `spec.originalUID` to match the new CR value `metadata.uid`. In this case, the restore will fail because the `spec.originalUID` in the saved (backed-up) CR does not match `spec.originalUID` in the new CR.
- The originalUID is only essential when the subsystem CR name exceeds 15 characters in length, or 10 characters limit for the API Connect Cluster CR. Recommended practice is that all backups should include the originalUID for Portal.
- See also [Preparing the Developer Portal for disaster recovery](#).

- g. Ensure that the endpoint definitions are the same as the previous Developer Portal CR.

2. Apply the Developer Portal subsystem custom resource (CR) YAML file (`portal_cr.yaml`) to the cluster as documented as part of the original installation instructions.

You can do this step now because the secrets are re-created, and additions made to `portal_cr.yaml`.

Note: Wait for the Developer Portal subsystem to become fully up and running before proceeding.

3. Use the following command to check that the Developer Portal database backups that are stored in the configured cloud storage are now present as Developer Portal backup CRs:

Note: You might wait for the cron job at the 10th minute `*/10 * * * *` to re-create your Developer Portal backup CRs.

```
kubectl get portalbackups -n <namespace>
```

For example:

```
loginabc@localhost [ibm-apiconnect]$ kubectl get pb
NAME          ID                STATUS  TYPE  CR TYPE  AGE   COMMENT
portal-bup-g66gv  20201001.100021  Ready  site  create   5m27s  y
portal-bup-z7wgk  20201001.100021  Ready  site  record   4m24s
```

The `AGE` of each is new.

4. Perform a Developer Portal restore. See [Backing up and restoring the Developer Portal in a Kubernetes environment](#)

For a disaster recovery scenario, it is advised that you restore all sites and systems. An example restore CR:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalRestore
metadata:
  generateName: portal-restore-
spec:
  type: all
  dryRun: false
  timestamp: now
  priorityList:
  - myportal-critical-site.com
  - myportal-important.com
```

5. When the Developer Portal restore CR is marked as `Ready`, the Disaster recovery is complete for the Developer Portal.

```
kubectl get portalrestore
```

6. Verify that you can access the restored Developer Portal sites and that you are able to log in to each one. The restore from the disaster event is now complete.

Recovering the Analytics subsystem after a disaster

Recover the API Connect Analytics subsystem in a Kubernetes environment.

About this task

Recovering the Analytics subsystem involves deploying a new installation of the Analytics subsystem, updating it with a saved copy of the Custom Resource (CR) that was originally used for installing Analytics, and then restoring the Analytics database from a back-up copy.

Procedure

1. Deploy the Analytics subsystem by applying the `analytics_cr.yaml` CR to the cluster as explained in [Installing analytics](#).
2. Restore your local copy of the original AnalyticsCluster CR by running the following command:

```
kubectl apply -f gj01-analytics.yaml
```

Replace `gj01-analytics.yaml` with the file name that you used for your saved CR.

3. Restore the Analytics data from a back up, as explained in [Restoring the Analytics database](#).

Disaster recovery on OpenShift and Cloud Pak for Integration

Prepare for, or recover from, a disaster affecting an API Connect deployment running on OpenShift or Cloud Pak for Integration

You must complete preparation steps prior to a disaster in order to recover API Connect.

- [Preparing for disaster recovery on OpenShift](#)
Back up data and essential deployment information from each API Connect subsystem so that you can use it to recover your deployment after a disaster.
- [Recovering from a disaster on OpenShift](#)
Install and configure API Connect, then restore backed up data to recover the deployment after a disaster.
- [Preparing for disaster recovery on Cloud Pak for Integration](#)
Back up data and essential deployment information from each API Connect subsystem so that you can use it to recover your deployment after a disaster.
- [Recovering from a disaster on Cloud Pak for Integration](#)
Install and configure API Connect, then restore the backed up data to recover the deployment after a disaster.

Preparing for disaster recovery on OpenShift

Back up data and essential deployment information from each API Connect subsystem so that you can use it to recover your deployment after a disaster.

About this task

To prepare your deployment for disaster, complete this task to copy the API Connect deployment settings, and then proceed to the following tasks to backup additional information for each subsystem.

Procedure

Complete the following steps to copy the API Connect deployment settings for use when you recover the deployment.

1. Find the API Connect cluster's name by running the following command and looking for the value in the NAME column:

```
oc get APIConnectCluster
```

The response looks like the following example:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
<code>apic-cluster</code>	<code>7/7</code>	<code>Running</code>	<code>10.0.3</code>	<code>10.0.3-149</code>		<code>6d4h</code>

2. Create a back-up copy of the installation CR for each subsystem, as well as the top-level APIConnectCluster CR, with the following commands:

```
a. oc -n <APIC_namespace> get apiconnectcluster apic-cluster-name -o yaml > apic-cluster-cr.yaml
```

```
b. oc -n <APIC_namespace> get mgmt apic-cluster-name-mgmt -o yaml > mgmt-susbsystem-cr.yaml
```

```
c. oc -n <APIC_namespace> get a7s apic-cluster-name-a7s -o yaml > a7s-susbsystem-cr.yaml
```

```
d. oc -n <APIC_namespace> get ptl apic-cluster-name-ptl -o yaml > ptl-susbsystem-cr.yaml
```

What to do next

Proceed to [Preparing the Management subsystem for disaster recovery on OpenShift](#).

1. [Preparing the Management subsystem for disaster recovery on OpenShift](#)
Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.
2. [Preparing the Portal subsystem for disaster recovery on OpenShift](#)
Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.
3. [Preparing the Analytics subsystem for disaster recovery on OpenShift](#)
Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

Preparing the Management subsystem for disaster recovery on OpenShift

Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

About this task

Make sure to store all backups and deployment information to a secure location where you can access it after a disaster. You cannot recover a deployment without the information that you collect during this task.

Procedure

1. Save a copy of the Management database encryption secret.
 - a. Run the following command to get the name of the encryption secret:

```
oc get mgmt -n <APIC-namespace> -o yaml | grep encryption
```

The output of this command shows the name of the secret that is storing the database encryption key, for example:

```
encryptionSecret: management-enc-key
```

- b. Run the following command to save a local copy of the secret as a YAML file:

```
oc get secret management-enc-key -n <APIC-namespace> -o yaml > management-enc-key.yaml
```

2. Save a copy of the Management client application credential secrets.

Save the credential secrets (the `client_id` and `client_secret`) for client applications provided by the Management subsystem.

- a. Run the following command to get the names of the secrets:

```
oc get mgmt -n <APIC-namespace> -o yaml | grep CredentialSecret
```

The output of this command shows the names of the secrets used to store each of the client application credentials. For example:

```
atmCredentialSecret: production-mgmt-atm-cred
consumerToolkitCredentialSecret: production-mgmt-ccli-cred
consumerUICredentialSecret: production-mgmt-cui-cred
designerCredentialSecret: production-mgmt-dsgr-cred
governanceCredentialSecret: production-mgmt-governance-cred
juhuCredentialSecret: production-mgmt-juhu-cred
toolkitCredentialSecret: production-mgmt-cli-cred
uiCredentialSecret: production-mgmt-ui-cred
```

- b. Run the following command to save a local copy of each secret as a YAML file, replacing `<secret_name>` with the name of the application credential secret:

```
oc get secret <secret_name> -n <APIC-namespace> -o yaml > <secret_name>.yaml
```

- c. Edit each of the saved YAML files and remove both the `ownerReferences` subsection and the `selfLink` property, and then save the updated file.

The information to remove looks like the following example:

```
ownerReferences:
- apiVersion: management.apiconnect.ibm.com/v1beta1
  blockOwnerDeletion: true
  controller: true
  kind: ManagementCluster
  name: management
  uid: 623e6b20-7eb8-46ce-94ac-6b64cd71afc4
selfLink: /api/v1/namespaces/default/secrets/management-atm-cred
```

3. Run a backup of the Management subsystem database.
 - a. [Configure the Management subsystem for backups.](#)
 - b. [Perform a backup of the Management subsystem.](#)
 - c. Get the name and ID of the newest Management backup and save them for reference during a recovery.

Run the following command to list the Management backups:

```
oc get mgmtb -n <APIC-namespace>
```

The response looks like the following example; the newest backup is the one with the smallest `AGE`:

NAME	STATUS	ID	CLUSTER	TYPE	CR TYPE	AGE
management-3aa3bebf	Ready	20200929-152150F	management	full	record	10h
management-3c4ca8df	Ready	20200929-115520F	management	full	record	20h
management-5tbxj	Ready	20200929-224349F	management	full	create	17h
management-f10af337	Ready	20200925-133703F	management	full	record	5d2h
management-jt1cd	Ready	20200929-224349F	management	full	create	25h
management-zxjtz	Ready	20200929-224349F	management	full	create	28h

4. Save a copy of the username and password for a Cloud Manager administrator.

When you recover the deployment, you will need to log in to the Cloud Manager as an administrator using the Cloud Manager User Registry. Any user with Administrator permission defined in the Cloud Manager User Registry can be used. Before backing up the information, verify that you can log in to Cloud Manager using that account.

If you want to backup the secret that contains the administrator password that was created by the API Connect operator during the original deployment, run the following command to get the secret:

```
oc --kubeconfig kubeconfig -n <APIC_namespace> get secret <instance-name>-mgmt-admin-pass -o jsonpath="{.data.password}" | base64 -d
```

It is possible that the password was changed at some point and the secret is now out-of-date. Be sure to verify that you can log in to Cloud Manager with the password contained in the secret. If the password is not valid, then make sure you have another Cloud Manager administrator account in the Cloud Manager User Registry, and back up that username and password instead.

5. Copy all of the information that you saved to a secure location.

What to do next

Immediately proceed to [Preparing the Portal subsystem for disaster recovery on OpenShift](#).

Important: Backups of the Management and Portal subsystems must be performed in sequence, and close together in time, to ensure that the Portal sites are consistent with Management database.

Next topic: [Preparing the Portal subsystem for disaster recovery on OpenShift](#)

Preparing the Portal subsystem for disaster recovery on OpenShift

Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

Before you begin

[Perform a backup of the Management subsystem](#).

Important: Backups of the Management and Portal subsystems must be performed in sequence, and close together in time, to ensure that the Portal sites are consistent with Management database.

About this task

Make sure to store all backups and deployment information to a secure location where you can access it after a disaster. You cannot recover a deployment without the information that you collect during this task.

Procedure

1. Save the ID of the Portal subsystem.

If your original CR `name` was 10 characters or longer, you must add the `originalUID` (Version 10.0.3 or later) or `metadata.uid` (Version 10.0.2 or earlier) value from the original Portal subsystem into the appropriate section of the installation CR during the recovery process.

When you deployed API Connect using the top-level `APIConnectCluster` CR, the deployment settings were added to subsystem-specific CRs that are managed by the top-level CR. When you recover the Portal subsystem after a disaster, you can locate the subsystem's ID in the saved copy of the Portal subsystem CR.

- a. Find the API Connect cluster's name by running the following command and looking for the value in the `NAME` column:

```
oc get APIConnectCluster
```

The response looks like the following example:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
<code>apic-cluster-name</code>	<code>7/7</code>	<code>Running</code>	<code>10.0.3</code>	<code>10.0.3-149</code>		<code>6d4h</code>

- b. Get the YAML that describes the Portal subsystem CR and save it in a file using the following command:

```
oc get ptl apic-cluster-name-ptl -o yaml > apic-cluster-name-ptl.yaml
```

2. Back up the Developer Portal database encryption secret.

- a. Run the following command to get the name of the encryption secret:

```
oc get ptl -n <APIC-namespace> -o yaml | grep encryption
```

The output of this command shows the name of the secret that is storing the database encryption key, for example:

```
encryptionSecret: portal-enc-key
```

- b. Run the following command to save a local copy of the secret as a YAML file:

```
oc get secret portal-enc-key -n <APIC-namespace> -o yaml > portal-enc-key.yaml
```

3. Back up the Portal endpoint definitions.

Save the information to a file because you must use the same names in the recovered deployment.

- a. Get the `portalAdminEndpoint` definition by running the following command:

```
oc get route | grep portal-director | awk '{print $2}'
```

The response looks like the following example:

```
dr-ptl-portal-director-apic.apps.vlxocp-1556.cp.fyre.ibm.com
```

- b. Get the `portalUIEndpoint` definition by running the following command:

```
oc get route | grep portal-web | awk '{print $2}'
```

The response looks like the following example:

```
dr-ptl-portal-web-apic.apps.vlxocp-1556.cp.fyre.ibm.com
```

4. Back up the Portal subsystem database.
 - a. [Configure the Portal subsystem for backups.](#)
 - b. [Perform a backup of the Portal subsystem.](#)

Note: Make sure that you set the property **type**: **all** when you backup the Portal database, to ensure that you capture data for the Portal subsystem and all Portal sites.

- c. Get the name of the newest Portal backup and save it for reference during a recovery.
Run the following command to list the Portal backups:

```
oc get portalbackup -n <APIC-namespace>
```

The response looks like the following example; the newest backup is the one with the smallest **AGE**:

NAME	ID	TASK ID	STATUS	TYPE	CR TYPE	CLUSTER	AGE	COMMENT
portal-manual-bup-91ghk	n/a	p5rkbbpyejrbqjab	Ready	all	create	production-pt1	15h	test comment
portal-manual-bup-jqbdm	n/a	cn8frkey1vgbls2u	Ready	all	create	production-pt1	15h	test comment

5. Copy all of the information that you saved to a secure location.

What to do next

Proceed to [Preparing the Analytics subsystem for disaster recovery on OpenShift.](#)

Previous topic: [Preparing the Management subsystem for disaster recovery on OpenShift](#)

Next topic: [Preparing the Analytics subsystem for disaster recovery on OpenShift](#)

Preparing the Analytics subsystem for disaster recovery on OpenShift

Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

About this task

Make sure to store all backups and deployment information to a secure location where you can access it after a disaster. You cannot recover a deployment without the information that you collect during this task.

Procedure

1. Backup the Analytics subsystem database.
 - a. [Configure the Analytics subsystem for backups.](#)
 - b. [Perform a backup of the Analytics subsystem.](#)

Note: Be sure to specify **indices**: **all** when you backup the Analytics database. Backing up a full set of indices ensures that your complete Analytics configuration, event data, and custom UI configuration (such as custom dashboards) can be restored.

- c. Get the name of the newest Analytics backup and save it for reference during a recovery.
Run the following command to list the Analytics backups:

```
oc get analyticsbackup -n <APIC-namespace>
```

The response looks like the following example; the newest backup is the one with the smallest **AGE**:

NAME	STATUS	ID	CR TYPE	INDICES	AGE	COMMENT
a7s-bup-266p8	Ready		create	[all]	20h	y
a7s-bup-pffdh	Ready		create	[all]	34h	y
analytics-bup-kxwq5	Ready	analytics-all-2020-09-07t03:49:15utc	record	[ui apievents config]	34h	
analytics-bup-p29z8	Ready	analytics-all-2020-09-07t18:04:03utc	record	[ui apievents config]	20h	

2. Copy all of the information that you saved to a secure location.

Previous topic: [Preparing the Portal subsystem for disaster recovery on OpenShift](#)

Recovering from a disaster on OpenShift

Install and configure API Connect, then restore backed up data to recover the deployment after a disaster.

Before you begin

To successfully recover API Connect from a disaster, you must have previously completed the steps to prepare the Management, Portal, and Analytics subsystems for disaster recovery. Make sure that you can access the backups and deployment information because you cannot recover your deployment without it.

Attention: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal.

About this task

To recover from a disaster event, recreate your API Connect Cluster CR and add recovery details as explained in the following procedure. Use the file containing the original CR (which you saved during the preparation task) as a reference.

Important: Do not install API Connect until you have configured the CR as explained in this task.

Procedure

1. Make a local copy of the `apiconnectcluster_cr.yaml` template custom resource so that you can customize it for your deployment. Template files are stored in `helper_files.zip`. For both OpenShift and CP4I, you can configure settings directly in the YAML file as explained in this task. In CP4I, you will then copy the YAML code and paste it in using the YAML tab in IBM Cloud Pak Platform UI.

2. Configure backup and restore settings for each subsystem.

Remember to create the backup secret for each subsystem as well, using the same name that you used in the original deployment (you can see the name in the `credentials` setting of each subsystem's backup section). The backup and restore settings, including the secrets, must exactly match the settings that you created in the original deployment. See the following topics for instructions on configuring backup and restore settings:

- [Configuring backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration](#)
- [Configuring backups for Developer Portal on OpenShift and Cloud Pak for Integration](#)
- [Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration](#)

Note:

During the disaster recovery process, the S3 configuration detail of the older management system is used, but the older management system must be in offline mode. The old subsystem must be offline because you cannot have two management systems simultaneously using the same s3 bucket name in the database backup configurations.

3. Apply the Management and Portal encryption secrets to the cluster.

a. Apply the YAML file that contains the Management database encryption secret into the cluster.

Run the following command, using the filename of the encryption secret that you saved when you prepared the Management subsystem for disaster recovery:

```
oc -n <APIC_namespace> apply -f mgmt-enc-key.yaml
```

b. Apply the YAML file that contains the Portal encryption secret into the cluster.

Run the following command, using the filename of the encryption secret that you saved when you prepared the Portal subsystem for disaster recovery:

```
oc -n <APIC_namespace> apply -f portal-enc-key.yaml
```

4. Important: Make sure that your new version of the API Connect Cluster installation CR uses the same value for the `name` setting as the original CR.

Edit the YAML file that you copied from the template in the `helper_files.zip`. In the following steps, you will configure additional deployment and recovery settings in the new API Connect Cluster installation CR before you install API Connect.

5. Add the Management and Portal encryption secrets to the API Connect Cluster CR.

Add each secret to the appropriate `spec.subsystem` section of the CR, as shown in the following example:

```
spec:
  management:
    encryptionSecret:
      secretName: mgmt-enc-key
  portal:
    encryptionSecret:
      secretName: portal-enc-key
```

6. Apply each of the Management client application credential secrets to the cluster.

Run the following command to apply each secret, using the filename of each client application credential secret that you saved when you prepared the Management subsystem for disaster recovery:

```
oc -n <APIC_namespace> apply -f <secretName>.yaml
```

For example, a deployment might use the following names for the secrets:

```
atmCredentialSecret: management-atm-cred
consumerToolkitCredentialSecret: management-ccli-cred
consumerUICredentialSecret: management-cui-cred
designerCredentialSecret: management-dsgr-cred
governanceCredentialSecret: management-governance-cred
juhuCredentialSecret: management-juhu-cred
toolkitCredentialSecret: management-cli-cred
uiCredentialSecret: management-ui-cred
```

7. Add the credential secrets to the `spec.management` section of API Connect Cluster CR.

For example:

```
spec:
  management:
    customApplicationCredentials:
      - name: atm-cred
        secretName: management-atm-cred
      - name: governance-cred
        secretName: management-governance-cred
      - name: ccli-cred
        secretName: management-ccli-cred
      - name: cui-cred
        secretName: management-cui-cred
      - name: dsgr-cred
        secretName: management-dsgr-cred
      - name: juhu-cred
        secretName: management-juhu-cred
      - name: cli-cred
        secretName: management-cli-cred
      - name: ui-cred
        secretName: management-ui-cred
```

8. Add the `siteName` property to the `spec.management` and `spec.portal` sections of the API Connect Cluster CR.
For example, if the `a2a5e6e2` is the original `siteName` of the Management subsystem, and `890772e3` is the original `siteName` of the Developer Portal subsystem, the CR looks like the following example:

```
spec:
  management:
    siteName: "a2a5e6e2"
  portal:
    siteName: "890772e3"
```

9. If your original CR `name` was 10 characters or longer, you must add the `originalUID` or `metadata.uid` value from the original Management and Portal subsystems into the appropriate sections of the installation CR.
Important: If your original CR name was 10 characters or longer and you do not add this value, the deployments and routes that are generated during installation will include a different UID, which will cause the recovery process to fail.
The IDs were created with the original deployment of the Management and Portal subsystems, and the recovered subsystems must continue to use the same IDs. You can locate the ID in the Management (`apic-cluster-name-mgt.yaml`) or Portal (`apic-cluster-name-ptl.yaml`) CR that you saved while preparing for disaster recovery. (Do not use the `metadata.uid` setting from the API Connect Cluster CR).

The CR setting that contains the ID depends on the version of API Connect that you are recovering:

- Version 10.0.3 or later: `spec.originalUID`
- Version 10.0.2 or earlier: `metadata.uid`

Locate each of the original settings in the appropriate subsystem CR, and copy them to corresponding sections of the new API Connect Cluster CR using the same key and value.

For example, if you originally deployed version 10.0.2, the setting name was `uid`. If you are deploying Version 10.0.3 as part of disaster recover, copy the original value and use it for the `originalUID` setting in the new CR. For example, in 10.0.3:

```
spec:
  management:
    originalUID: "fa0f6f49-b931-4472-b84d-0922a9a92dfd"
  portal:
    originalUID: "447ea4b3-9514-4a84-a34b-ce0b349838da"
```

10. If you installed the API Connect using an S3 provider for the Management subsystem backups, add the following annotation to the `apiconnectcluster` CR. If you specified SFTP or local as the backup, skip this step.

- Using the Platform UI:
 - Edit the API Management instance and click the YAML tab to edit the CR.
 - Add the following statement to the `spec.metadata.annotations` section:

```
apiconnect-operator/deployment-mode: disasterRecovery
```

For example:

```
metadata:
  annotations:
    apiconnect-operator/deployment-mode: disasterRecovery
```

- Using the CLI:
 - Get the name of the CR by running the following command:

```
oc get apiconnectcluster -o name -n <APIC_namespace>
```

The response looks like the following example:

```
apiconnectcluster.apiconnect.domain/instance_name
```

- Add the following annotation to the `spec.metadata.annotations` section of the CR by running the following command:

```
oc annotate apiconnectcluster/instance_name apiconnect-operator/deployment-mode="disasterRecovery" -n <APIC_namespace>
```

The response looks like the following example:

```
apiconnectcluster.apiconnect.domain/instance_name annotated
```

11. Verify that you correctly added all of the settings to the API Connect Cluster CR.
For example, the completed CR might look like the following example:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APICConnectCluster
metadata:
  namespace: apiconnect
  name: apis-minimum
  labels:
    app.kubernetes.io/instance: apiconnect
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: apiconnect-minimum
  annotations:
    apiconnect-operator/deployment-mode: disasterRecovery
spec:
  license:
    accept: true
    license: L-RJON-C2YLGB
    metric: VIRTUAL_PROCESSOR_CORE
    use: nonproduction
  management:
    originalUID: "fa0f6f49-b931-4472-b84d-0922a9a92dfd"
    encryptionSecret:
      secretName: apis-minim-faaa44bf-enc-key
```

```

siteName: "faaa44bf"
customApplicationCredentials:
  - name: atm-cred
  - secretName: apis-minim-faaa44bf-atm-cred
  - name: governance-cred
  - secretName: management-governance-cred
  - name: ccli-cred
  - secretName: apis-minim-faaa44bf-ccli-cred
  - name: cli-cred
  - secretName: apis-minim-faaa44bf-cli-cred
  - name: cui-cred
  - secretName: apis-minim-faaa44bf-cui-cred
  - name: dsgr-cred
  - secretName: apis-minim-faaa44bf-dsgr-cred
  - name: juhu-cred
  - secretName: apis-minim-faaa44bf-juhu-cred
  - name: ui-cred
  - secretName: apis-minim-faaa44bf-ui-cred
analytics:
  client: {}
  ingestion: {}
apiManagerEndpoint: {}
cloudManagerEndpoint: {}
consumerAPIEndpoint: {}
platformAPIEndpoint: {}
portal:
  admin: {}
databaseBackup:
  path: ocp-dr-mgmt
  host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
  s3provider: ibm
  schedule: 0 1 * * *
  protocol: objstore
  credentials: mgmt-backup-secret
analytics:
  storage:
    enabled: true
    type: unique
  databaseBackup:
    chunkSize: 1GB
    credentials: a7s-backup-secret
    host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
    path: ocp-dr-a7s
    schedule: 0 1 * * *
storageClassName: rook-ceph-block
profile: nlxc7.m48
portal:
  adminClientSubjectDN: ""
  portalAdminEndpoint: {}
  portalUIEndpoint: {}
  originalUID: "447ea4b3-9514-4a84-a34b-ce0b349838da"
  encryptionSecret:
    secretName: apis-minim-913edd20-enc-key
  siteName: "913edd20"
  portalBackup:
    credentials: portal-backup-secret
    host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
    path: ocp-dr-portal
    protocol: objstore
    schedule: 0 1 * * *
version: 10.0.3

```

12. Prepare your environment for installing API Connect by completing the prerequisites and all components of step 1 in [Installing API Connect](#). After you prepare your environment, continue with this procedure to install API Connect and then restore data.

13. Install API Connect by creating the CR with the following command:
Replace `<cr_name>` with the value of the `name` setting in the API Connect Cluster CR.

```
oc -n <APIC_namespace> create -f <cr_name>.yaml
```

14. Wait for all subsystems to be created.
When the Management subsystem is installed, you might see `backup` job pods and `stanza-create` job pods in `Error` state; for example:

m1-82b290a2-postgres-stanza-create-4zcgz	0/1	Error	0	35m
m1-82b290a2-postgres-full1-sch-backup-2g9hm	0/1	Error	0	20m

This is expected behavior, for the following reasons:

- The `stanza-create` job normally expects buckets or subdirectories within buckets to be empty. However, since you configured the Management subsystem with your already-populated S3 bucket (where your backups exist), the job will enter the `Error` state.
- Any scheduled or manual backups will enter the `Error` state. Although you configured the Management subsystem with your already-populated S3 bucket, the new database isn't yet configured to write backups into remote storage.

The errors will not prevent a successful restore, so continue to the next step.

Important:

For S3, the recovery remains in an intermediate state until the restore is complete, and Postgres wal files might cause serious disk issues. To avoid this possibility, continue immediately with the next step.

Note that if you delay completion of the restore:

- Health check might fail. In this case, you can still proceed to the next step and perform a restore.
- Postgres wal files might cause problems by consuming all disk space. In this case, you must either:
 - Re-install the system, prepare again for disaster recovery, and perform the restore.

- Or increase disk space so that the system returns to a stable state, and then proceed with the restore.
- Restore the subsystems in the following sequence:
 - [Restore the Management subsystem.](#)
When you perform a restore, you must complete the restoration of the Management subsystem first, and then immediately restore the Portal subsystem.

Important: Be sure to restore the Management subsystem from the backup whose name and ID of the you noted while preparing for disaster recovery.
 - [Restore the Developer Portal subsystem.](#)
Use the restore type `a11` to ensure that you restore the complete subsystem and all Portal sites.
 - [Restore the Analytics subsystem.](#)
 - Version 10.0.2 or earlier: Update the Management OIDC credentials as explained in [Configuring the OIDC credentials on OpenShift](#).
 - Verify that the recovery was successful:
 - Ensure that you can log in to the Cloud Manager UI.
 - Verify that your provider organizations exist.
 - Ensure that you can log in to each Developer portal.
 - Ensure that the Analytics dashboard contains all of the analytics data that you preserved.
 - After the successful recovery, remove the annotation that you added to the `apiconnectcluster` CR in step [10](#).
If you skipped step [10](#), then skip this step as well.
 - Using the Platform UI:
 - Edit the API Management instance and click the YAML tab to edit the CR.
 - Delete the following statement from the `spec.metadata.annotations` section:


```
apiconnect-operator/deployment-mode: disasterRecovery
```
 - Using the CLI:
 - Get the name of the CR by running the following command:


```
oc get apiconnectcluster -o name -n <APIC_namespace>
```

 The response looks like the following example:


```
apiconnectcluster.apiconnect.domain/instance_name
```
 - Remove the annotation by running the following command, making sure to include the trailing "-" on `deployment-mode-` to indicate the removal:


```
oc -n <APIC_namespace> annotate apiconnectcluster/instance_name apiconnect-operator/deployment-mode-
```

 The response looks like the following example:


```
apiconnectcluster.apiconnect.domain/instance_name annotated
```
 - [Configuring the OIDC credentials on OpenShift](#)
When you recover API Connect V10.0.x, you must manually update the values for the Management OIDC credentials (`client_id` and `client_secret`) in the Common Services User Registry.

Configuring the OIDC credentials on OpenShift

When you recover API Connect V10.0.x, you must manually update the values for the Management OIDC credentials (`client_id` and `client_secret`) in the Common Services User Registry.

Before you begin

This task is required for all of the 10.0.x releases up to, and including, version 10.0.2 (skip this task for version 10.0.3 and later). The procedure requires you to log in to the Cloud Manager on the new API Connect deployment. Use the Cloud Manager administrator username and password that you saved while preparing for disaster recovery.

About this task

When you deploy API Connect, the OIDC `client_id` and `client_secret` resources are automatically generated and added to the Common Services User Registry. These resources enable single sign-on with the IBM Cloud Pak platform. When you reinstall API Connect as part of disaster recovery, the original versions of those resources are retained but cannot be used. You must update the Common Services User Registry and replace the original values with the values from the new deployment.

Procedure

- Retrieve the `client_id` and `client_secret` from the stored secret in API Connect.
 - Run the following command to retrieve the `client_id`:


```
oc -n <APIC_namespace> get secret <instance_name>-oidc-client -o jsonpath="{.data.CLIENT_ID}" | base64 -D
```
 - Run the following command to retrieve the `client_secret`:


```
oc -n <APIC_namespace> get secret <instance_name>-oidc-client -o jsonpath="{.data.CLIENT_SECRET}" | base64 -D
```
- Update the Common Services User Registry with the new values.
Using the Cloud Manager UI:
 - In API Connect, open the Cloud Manager interface.
 - Select the Cloud Manager User Registry.
You will use the Cloud Manager User Registry to modify settings for the Common Service User Registry.

- c. Log in to the Cloud Manager User Registry using the administrator username and password that you saved while preparing for disaster recovery.
- d. In Cloud Manager, click Resources, User Registries and edit Common Services User Registry.
- e. In the Client information section, paste the `client_ID` value from step 1 into the Client ID field.
- f. Paste the `client_secret` value from step 1 into the Client secret field.
- g. Save your changes.

Using the toolkit CLI:

- a. Run the following command to determine the `<mgmt_endpoint_URL>` that you will use for accessing the management server:

```
oc -n <APIC_namespace> get mgmt <instance_name> -o jsonpath="{.status.zenRoute}" && echo ""
```

- b. Log in to the management server:

- i. Run the following command to start the login process:

```
apic login --server <mgmt_endpoint_URL>
```

- ii. Provide your login credentials:

- **Realm?** Type your realm (for example: `admin/default-idp-1`) and press Enter.
The realm refers to your identity provider. For information on determining your realm value, see [How to determine the identity provider](#).
- **Username?** Type your Cloud Manager administrator username and press Enter.
- **Password?** Type the password for your Cloud Manager administrator account and press Enter.

- c. Run the following command to download the Common Service User Registry as a YAML file:

```
apic user-registries:get common-services --server <mgmt_endpoint_URL> -o admin --fields name,configuration
```

- d. Edit the file and update the `client_ID` and `client_secret` settings with the values you obtained in step 1.

- e. Save and close the file.

- f. Run the following command to upload the modified file to Cloud Manager:

```
apic user-registries:update common-services --server <mgmt_endpoint_URL> -o admin common-services.yaml
```

Preparing for disaster recovery on Cloud Pak for Integration

Back up data and essential deployment information from each API Connect subsystem so that you can use it to recover your deployment after a disaster.

About this task

To prepare your deployment for disaster, complete this task to copy the API Connect deployment settings, and then proceed to the following tasks to backup additional information for each subsystem.

Procedure

Complete the following steps to copy the API Connect deployment settings for use when you recover the deployment.

1. Find the API Connect cluster's name by running the following command and looking for the value in the NAME column:

```
oc get APICConnectCluster
```

The response looks like the following example:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
apic-cluster	7/7	Running	10.0.3	10.0.3-149		6d4h

2. Create a back-up copy of the installation CR for each subsystem, as well as the top-level APICConnectCluster CR, with the following commands:

```
a. oc -n <APIC_namespace> get apicconnectcluster <apic-cluster-name> -o yaml > apic-cluster-cr.yaml
```

```
b. oc -n <APIC_namespace> get mgmt <apic-cluster-name>-mgmt -o yaml > mgmt-subsytem-cr.yaml
```

```
c. oc -n <APIC_namespace> get a7s <apic-cluster-name>-a7s -o yaml > a7s-subsytem-cr.yaml
```

```
d. oc -n <APIC_namespace> get ptl <apic-cluster-name>-ptl -o yaml > ptl-subsytem-cr.yaml
```

What to do next

Proceed to [Preparing the Management subsystem for disaster recovery on Cloud Pak for Integration](#).

1. [Preparing the Management subsystem for disaster recovery on Cloud Pak for Integration](#)
Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.
2. [Preparing the Portal subsystem for disaster recovery on Cloud Pak for Integration](#)
Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.
3. [Preparing the Analytics subsystem for disaster recovery on Cloud Pak for Integration](#)
Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

Preparing the Management subsystem for disaster recovery on Cloud Pak for Integration

Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

About this task

Make sure to store all backups and deployment information to a secure location where you can access it after a disaster. You cannot recover a deployment without the information that you collect during this task.

Procedure

1. Save a copy of the Management database encryption secret.

- a. Run the following command to get the name of the encryption secret:

```
oc get mgmt -n <APIC-namespace> -o yaml | grep encryption
```

The output of this command shows the name of the secret that is storing the database encryption key; for example:

```
encryptionSecret: management-enc-key
```

- b. Run the following command to save a local copy of the secret as a YAML file:

```
oc get secret management-enc-key -n <APIC-namespace> -o yaml > management-enc-key.yaml
```

2. Save a copy of the Management client application credential secrets.

Save the credential secrets (the `client_id` and `client_secret`) for client applications provided by the Management subsystem.

- a. Run the following command to get the names of the secrets:

```
oc get mgmt -n <APIC-namespace> -o yaml | grep CredentialSecret
```

The output of this command shows the names of the secrets used to store each of the client application credentials; for example:

```
atmCredentialSecret: management-atm-cred
consumerToolkitCredentialSecret: management-ccli-cred
consumerUICredentialSecret: management-cui-cred
designerCredentialSecret: management-dsgr-cred
governanceCredentialSecret: management-governance-cred
juhuCredentialSecret: management-juhu-cred
toolkitCredentialSecret: management-cli-cred
uiCredentialSecret: management-ui-cred
```

- b. Run the following command to save a local copy of each secret as a YAML file, replacing `<secret_name>` with the name of the application credential secret:

```
oc get secret <secret_name> -n <APIC-namespace> -o yaml > <secret_name>.yaml
```

- c. Edit each of the saved YAML files and remove both the `ownerReferences` subsection and the `selfLink` property, and then save the updated file.

The information to remove looks like the following example:

```
ownerReferences:
- apiVersion: management.apiconnect.ibm.com/v1beta1
  blockOwnerDeletion: true
  controller: true
  kind: ManagementCluster
  name: management
  uid: 623e6b20-7eb8-46ce-94ac-6b64cd71afc4
  selfLink: /api/v1/namespaces/default/secrets/management-atm-cred
```

3. Save a copy of the CP4I credentials secret.

- a. Run the following command to get the name of the encryption secret and save a local copy of the secret as a YAML file:

```
oc get secrets <instance_name>-cp4i-creds -o yaml > <cp4i_creds_secret>.yaml
```

- b. Edit the saved YAML file and remove both the `ownerReferences` subsection and the `selfLink` property, and then save the updated file.

4. Run a backup of the Management subsystem database.

- a. [Configure the Management subsystem for backups.](#)

- b. [Perform a backup of the Management subsystem.](#)

- c. Get the name and ID of the newest Management backup and save them for reference during a recovery.

Run the following command to list the Management backups:

```
oc get mgmtb -n <APIC-namespace>
```

The response looks like the following example; the newest backup is the one with the smallest `AGE`:

NAME	STATUS	ID	CLUSTER	TYPE	CR TYPE	AGE
management-3aa3bebf	Ready	20200929-152150F	management	full	record	10h
management-3c4ca8df	Ready	20200929-115520F	management	full	record	20h
management-5tbxj	Ready	20200929-224349F	management	full	create	17h
management-f10af337	Ready	20200925-133703F	management	full	record	5d2h
management-jt1cd	Ready	20200929-224349F	management	full	create	25h
management-zxjtz	Ready	20200929-224349F	management	full	create	28h

5. Save a copy of the username and password for a Cloud Manager administrator.

When you recover the deployment, you will need to log in to the Cloud Manager as an administrator using the Cloud Manager User Registry. Any user with Administrator permission defined in the Cloud Manager User Registry can be used. Before backing up the information, verify that you can log in to Cloud Manager using that account.

If you want to backup the secret that contains the administrator password that was created by the API Connect operator during the original deployment, run the following command to get the secret:

```
oc -n <APIC_namespace> get secret <instanceName>-mgmt-admin-pass -o jsonpath="{.data.password}" | base64 -d
```

It is possible that the password was changed at some point and the secret is now out-of-date. Be sure to verify that you can log in to Cloud Manager with the password contained in the secret. If the password is not valid, then make sure you have another Cloud Manager administrator account in the Cloud Manager User Registry, and back up that username and password instead.

6. Copy all of the information that you saved to a secure location.

What to do next

Immediately proceed to [Preparing the Portal subsystem for disaster recovery on Cloud Pak for Integration](#).

Important: Backups of the Management and Portal subsystems must be performed in sequence, and close together in time, to ensure that the Portal sites are consistent with Management database.

Next topic: [Preparing the Portal subsystem for disaster recovery on Cloud Pak for Integration](#)

Preparing the Portal subsystem for disaster recovery on Cloud Pak for Integration

Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

Before you begin

[Perform a backup of the Management subsystem](#).

Important: Backups of the Management and Portal subsystems must be performed in sequence, and close together in time, to ensure that the Portal sites are consistent with Management database.

About this task

Make sure to store all backups and deployment information to a secure location where you can access it after a disaster. You cannot recover a deployment without the information that you collect during this task.

Procedure

1. Save the ID of the Portal subsystem.

If your original CR `name` was 10 characters or longer, you must add the `originalUID` (Version 10.0.3 or later) or `metadata.uid` (Version 10.0.2 or earlier) value from the original Portal subsystem into the appropriate section of the installation CR during the recovery process.

When you deployed API Connect using the top-level `APICConnectCluster` CR, the deployment settings were added to subsystem-specific CRs that are managed by the top-level CR. When you recover the Portal subsystem after a disaster, you can locate the subsystem's ID in the saved copy of the Portal subsystem CR.

- a. Find the API Connect cluster's name by running the following command and looking for the value in the `NAME` column:

```
oc get APICConnectCluster
```

The response looks like the following example:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
<code>apic-cluster-name</code>	<code>7/7</code>	<code>Running</code>	<code>10.0.3</code>	<code>10.0.3-149</code>		<code>6d4h</code>

- b. Get the YAML that describes the Portal subsystem CR and save it in a file using the following command:

```
oc get ptl apic-cluster-name-ptl -o yaml > apic-cluster-name-ptl.yaml
```

2. Back up the Developer Portal database encryption secret.

- a. Run the following command to get the name of the encryption secret:

```
oc get ptl -n <APIC-namespace> -o yaml | grep encryption
```

The output of this command shows the name of the secret that is storing the database encryption key, for example:

```
encryptionSecret: portal-enc-key
```

- b. Run the following command to save a local copy of the secret as a YAML file:

```
oc get secret portal-enc-key -n <APIC-namespace> -o yaml > portal-enc-key.yaml
```

3. Back up the Portal endpoint definitions.

Save the information to a file because you must use the same names in the recovered deployment.

- a. Get the `portalAdminEndpoint` definition by running the following command:

```
oc get route | grep portal-director | awk '{print $2}'
```

The response looks like the following example:

```
dr-ptl-portal-director-apic.apps.vlxocp-1556.cp.fyre.ibm.com
```

- b. Get the `portalUIEndpoint` definition by running the following command:

```
oc get route | grep portal-web | awk '{print $2}'
```

The response looks like the following example:

```
dr-ptl-portal-web-apic.apps.vlxocp-1556.cp.fyre.ibm.com
```

4. Back up the Portal subsystem database.
 - a. [Configure the Portal subsystem for backups.](#)
 - b. [Perform a backup of the Portal subsystem.](#)

Note: Be sure to **type**: **all** when you backup the Portal database, to ensure that you capture data for the Portal subsystem and all Portal sites.
- c. Get the ID of the latest **all** Portal backup and save it for reference during a recovery.
Run the following command to list the Portal backups:

```
oc get portalbackup -n <APIC-namespace>
```

The response looks like the following example; the latest backup is the one with a **TYPE** of **all** and the smallest **AGE**:

NAME	ID	STATUS	TYPE	CR TYPE	AGE	COMMENT
p1-bup-gqz8f	20200526.142449	Ready	system	record	55m	
p1-bup-jx4vc	20200526.142458	Ready	site	record	54m	
p1-bup-st412		Ready	all	create	56m	y
p1-bup-th2dm		Ready	all	create	4h51m	y

5. Copy all of the information that you saved to a secure location.

What to do next

Proceed to [Preparing the Analytics subsystem for disaster recovery on Cloud Pak for Integration.](#)

Previous topic: [Preparing the Management subsystem for disaster recovery on Cloud Pak for Integration](#)

Next topic: [Preparing the Analytics subsystem for disaster recovery on Cloud Pak for Integration](#)

Preparing the Analytics subsystem for disaster recovery on Cloud Pak for Integration

Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

About this task

Make sure to store all backups and deployment information to a secure location where you can access it after a disaster. You cannot recover a deployment without the information that you collect during this task.

Procedure

1. Backup the Analytics subsystem database.
 - a. [Configure the Analytics subsystem for backups.](#)
 - b. [Perform a backup of the Analytics subsystem.](#)

Note: Be sure to specify **indices**: **all** when you backup the Analytics database. Backing up a full set of indices ensures that your complete Analytics configuration, event data, and custom UI configuration (such as custom dashboards) can be restored.
- c. Get the name of the newest Analytics backup and save it for reference during a recovery.
Run the following command to list the Analytics backups:

```
oc get analyticsbackup -n <APIC-namespace>
```

The response looks like the following example; the newest backup is the one with the smallest **AGE**:

NAME	STATUS	ID	CR TYPE	INDICES	AGE	COMMENT
a7s-bup-266p8	Ready		create	[all]	20h	y
a7s-bup-pffdh	Ready		create	[all]	34h	y
analytics-bup-kxwq5	Ready	analytics-all-2020-09-07t03:49:15utc	record	[ui apievents config]	34h	
analytics-bup-p29z8	Ready	analytics-all-2020-09-07t18:04:03utc	record	[ui apievents config]	20h	

2. Copy all of the information that you saved to a secure location.

Previous topic: [Preparing the Portal subsystem for disaster recovery on Cloud Pak for Integration](#)

Recovering from a disaster on Cloud Pak for Integration

Install and configure API Connect, then restore the backed up data to recover the deployment after a disaster.

Before you begin

To successfully recover API Connect from a disaster, you must have previously completed the steps to prepare the Management, Portal, and Analytics subsystems for disaster recovery. Make sure that you can access the backups and deployment information because you cannot recover your deployment without it.

Attention: Successful disaster recovery depends on recovery of both the Management subsystem and the Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management subsystem first, and then immediately restore the Portal subsystem.

About this task

To recover from a disaster event, recreate your API Connect Cluster CR and add recovery details as explained in the following procedure. Use the file containing the original CR (which you saved during the preparation task) as a reference.

Note: Your recovery installation of Cloud Pak for Integration must include the Platform UI if your original deployment used it. In addition, the CR must use the same settings as in the original deployment.

Procedure

1. Prepare your Cloud Pak for Integration environment for deploying API Connect as explained in the appropriate version of the [Cloud Pak for Integration documentation](#).

After you prepare your environment, return to this procedure to configure the installation CR, deploy API Connect, and then restore data to the API Connect subsystems.

Important: Do not install API Connect until you have configured the CR as explained in following steps.

2. Apply the Management and Portal encryption secrets to the cluster where you will install API Connect.

- a. Apply the YAML file that contains the Management database encryption secret into the cluster.

Run the following command, using the filename of the encryption secret that you saved when you prepared the Management subsystem for disaster recovery:

```
oc apply -f mgmt-enc-key.yaml
```

- b. Apply the YAML file that contains the Portal encryption secret into the cluster.

Run the following command, using the filename of the encryption secret that you saved when you prepared the Portal subsystem for disaster recovery:

```
oc apply -f portal-enc-key.yaml
```

3. Apply the YAML file that contains the CP4I credentials secret into the cluster.

Run the following command, using the filename of the secret that you saved when you prepared the Management subsystem for disaster recovery:

```
oc apply -f <cp4i_creds_secret>.yaml -n <APIC_namespace>
```

4. Set up the installation CR for deploying API Connect.

In this step, you perform the initial steps for deploying API Connect in Cloud Pak for Integration, but you do not begin the actual installation until step [15](#). Instead, you will complete a series of steps to configure the CR with deployment settings and additional recovery settings.

- a. Log in to the IBM Cloud Pak Platform UI.
- b. On the home page, click Create capability.
- c. Select the API Connect tile and click Next.
- d. On the Create an instance of API Connect cluster page, select the deployment type and click Next.
Choose the same deployment type that you used when you originally deployed API Connect.

- e. On the deployment settings page, click the YAML tab to edit the installation CR in YAML format.

- f. Copy the content of the saved CR into the YAML tab, replacing the default CR.

Keep the YAML tab open. In the following steps, you will configure additional deployment and recovery settings in the new API Connect Cluster installation CR before you install API Connect.

5. Important: Make sure that your new version of the API Connect Cluster installation CR uses the same value for the **name** setting as the original CR.

6. Verify the backup and restore settings for each subsystem.

For more information on the backup settings, see the following topics for instructions on configuring backup and restore settings:

- [Configuring backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration](#)
- [Configuring backups for Developer Portal on OpenShift and Cloud Pak for Integration](#)
- [Configuring backup settings for Analytics on OpenShift and Cloud Pak for Integration](#)

7. Generate the Kubernetes secret for each subsystem's backups by running the appropriate command:

For each secret, use the same name that you used in the original deployment (you can see the name in the **credentials** setting of each subsystem's backup section).

- Management subsystem:

- S3:

```
oc create secret generic mgmt-backup-secret --from-literal=username='<Your_access_key-or-user_name>' --from-literal=password='<Your_access_key_secret-or-password>' -n <APIC_namespace>
```

- SFTP with username and password:

```
oc create secret generic mgmt-backup-secret --from-literal=username='<Your_user_name>' --from-literal=password='<Your_password>' -n <APIC_namespace>
```

- SFTP with username and SSH-key

```
oc create secret generic mgmt-backup-secret --from-literal=username='<Your_user_name>' --from-file=ssh-privatekey='<Your_private_key_file>' -n <APIC_namespace>
```

- Portal subsystem:

- S3:

```
oc create secret generic portal-backup-secret --from-literal=username='<Your_access_key-or-user_name>' --from-literal=password='<Your_access_key_secret-or-password>' -n <APIC_namespace>
```

- SFTP with username and password:

```
oc create secret generic portal-backup-secret --from-literal=username='<Your_user_name>' --from-literal=password='<Your_password>' -n <APIC_namespace>
```

- SFTP with username and SSH-key

```
oc create secret generic portal-backup-secret --from-literal=username='<Your_user_name>' --from-file=ssh-privatekey='<Your_private_key_file>' -n <APIC_namespace>
```

- Analytics:

```
oc create secret generic analytics-backup-secret --from-literal=access_key='<Your_Access_Key>' --from-literal=secret_key='<Your_access_key_secret>' -n <APIC_namespace>
```

8. Add the saved Management and Portal encryption secrets to the installation CR.
Add each secret to the appropriate `spec.subsystem` section of the CR, as shown in the following example:

```
spec:
  management:
    encryptionSecret:
      secretName: mgmt-enc-key
  portal:
    encryptionSecret:
      secretName: portal-enc-key
```

9. Apply each of the Management client application credential secrets to the cluster.
Run the following command to apply each secret, using the filename of each client application credential secret that you saved when you prepared the Management subsystem for disaster recovery:

```
oc apply -f <secretName>.yaml
```

For example, a deployment might use the following names for the secrets:

```
atmCredentialSecret: management-atm-cred
consumerToolkitCredentialSecret: management-ccli-cred
consumerUICredentialSecret: management-cui-cred
designerCredentialSecret: management-dsgr-cred
governanceCredentialSecret: management-governance-cred
juhuCredentialSecret: management-juhu-cred
toolkitCredentialSecret: management-cli-cred
uiCredentialSecret: management-ui-cred
```

10. Add each of the credential secrets to the `spec.management` section of installation CR.
For example:

```
spec:
  management:
    customApplicationCredentials:
      - name: atm-cred
        secretName: management-atm-cred
      - name: ccli-cred
        secretName: management-ccli-cred
      - name: cui-cred
        secretName: management-cui-cred
      - name: dsgr-cred
        secretName: management-dsgr-cred
      - name: governance-cred
        secretName: management-governance-cred
      - name: juhu-cred
        secretName: management-juhu-cred
      - name: cli-cred
        secretName: management-cli-cred
      - name: ui-cred
        secretName: management-ui-cred
```

11. Add the `siteName` property to the `spec.management` and `spec.portal` sections of the installation CR.
For example, if the `a2a5e6e2` is the original `siteName` of the Management subsystem, and `890772e3` is the original `siteName` of the Portal subsystem, the CR looks like the following example:

```
spec:
  management:
    siteName: "a2a5e6e2"
  portal:
    siteName: "890772e3"
```

12. If your original CR `name` was 10 characters or longer, you must add the `originalUID` or `metadata.uid` value from the original Management and Portal subsystems into the appropriate sections of the installation CR.
Important: If your original CR name was 10 characters or longer and you do not add this value, the deployments and routes that are generated during installation will include a different UID, which will cause the recovery process to fail.
The IDs were created with the original deployment of the Management and Portal subsystems, and the recovered subsystems must continue to use the same IDs. You can locate the ID in the Management (`apic-cluster-name-mgt.yaml`) or Portal (`apic-cluster-name-ptl.yaml`) CR that you saved while preparing for disaster recovery. (Do not use the `metadata.uid` setting from the API Connect Cluster CR).

The CR setting that contains the ID depends on the version of API Connect that you are recovering:

- Version 10.0.3 or later: `spec.originalUID`
- Version 10.0.2 or earlier: `metadata.uid`

Locate each of the original settings in the appropriate subsystem CR, and copy them to corresponding sections of the new API Connect Cluster CR using the same key and value.

For example, if you originally deployed version 10.0.2, the setting name was `uid`. If you are deploying Version 10.0.3 as part of disaster recover, copy the original value and use it for the `originalUID` setting in the new CR. For example, in 10.0.3:

```
spec:
  management:
    originalUID: "fa0f6f49-b931-4472-b84d-0922a9a92dfd"
  portal:
    originalUID: "447ea4b3-9514-4a84-a34b-ce0b349838da"
```

13. If you installed the API Connect using an S3 provider for the Management subsystem backups, add the following annotation to the `apiconnectcluster` CR. If you specified SFTP or local as the backup, skip this step.

- Using the Platform UI:
 - Edit the API Management instance and click the YAML tab to edit the CR.
 - Add the following statement to the `spec.metadata.annotations` section:

```
apiconnect-operator/deployment-mode: disasterRecovery
```

For example:

```
metadata:
  annotations:
    apiconnect-operator/deployment-mode: disasterRecovery
```

- Using the CLI:
 - Get the name of the CR by running the following command:

```
oc get apiconnectcluster -o name -n <APIC_namespace>
```

The response looks like the following example:

```
apiconnectcluster.apiconnect.domain/instance_name
```

- Add the following annotation to the `spec.metadata.annotations` section of the CR by running the following command:

```
oc annotate apiconnectcluster/instance_name apiconnect-operator/deployment-mode="disasterRecovery" -n <APIC_namespace>
```

The response looks like the following example:

```
apiconnectcluster.apiconnect.domain/instance_name annotated
```

14. Verify that you correctly added all of the settings to the installation CR.

For example, the completed CR might look like the following example:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APICConnectCluster
metadata:
  namespace: apiconnect
  name: apis-minimum
  labels:
    app.kubernetes.io/instance: apiconnect
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: apiconnect-minimum
spec:
  license:
    accept: true
    license: L-RJON-C2YLG
    metric: VIRTUAL_PROCESSOR_CORE
    use: nonproduction
  management:
    originalUID: "fa0f6f49-b931-4472-b84d-0922a9a92dfd"
    encryptionSecret:
      secretName: apis-minim-faaa44bf-enc-key
    siteName: "faaa44bf"
    customApplicationCredentials:
      - name: atm-cred
        secretName: apis-minim-faaa44bf-atm-cred
      - name: cli-cred
        secretName: apis-minim-faaa44bf-cli-cred
      - name: cli-cred
        secretName: apis-minim-faaa44bf-cli-cred
      - name: cui-cred
        secretName: apis-minim-faaa44bf-cui-cred
      - name: dsgr-cred
        secretName: apis-minim-faaa44bf-dsgr-cred
      - name: governance-cred
        secretName: apis-minim-faaa44bf-governance-cred
      - name: juhu-cred
        secretName: apis-minim-faaa44bf-juhu-cred
      - name: ui-cred
        secretName: apis-minim-faaa44bf-ui-cred
  analytics:
    client: {}
    ingestion: {}
  apiManagerEndpoint: {}
  cloudManagerEndpoint: {}
  consumerAPIEndpoint: {}
  platformAPIEndpoint: {}
  portal:
    admin: {}
  databaseBackup:
    path: ocp-dr-mgmt
    host: s3.eu-gb.cloud-object-storage.apptdomain.cloud/eu-standard
    s3provider: ibm
    schedule: 0 1 * * *
    protocol: objstore
    credentials: mgmt-backup-secret
  analytics:
    storage:
      enabled: true
      type: unique
  databaseBackup:
```

```

chunkSize: 1GB
credentials: a7s-backup-secret
host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
path: ocp-dr-a7s
schedule: 0 1 * * *
storageClassName: rook-ceph-block
profile: nlxc7.m48
portal:
  adminClientSubjectDN: ""
  portalAdminEndpoint: {}
  portalUIEndpoint: {}
  originalUID: "447ea4b3-9514-4a84-a34b-ce0b349838da"
  encryptionSecret:
    secretName: apis-minim-913edd20-enc-key
  siteName: "913edd20"
  portalBackup:
    credentials: portal-backup-secret
    host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
    path: ocp-dr-portal
    protocol: objstore
    schedule: 0 1 * * *
version: 10.0.3

```

Important: If you are restoring your deployment in a different data center, the endpoints that are used in your original deployment must be the same in your new data center. The Platform UI in Cloud Pak for Integration autogenerates the endpoints if they are left empty in the CR, therefore ensure that you explicitly set the endpoints in the CR to match what was set in your original data center. For example:

```

apiManagerEndpoint:
  annotations:
    cert-manager.io/issuer: prod-ingress-issuer
    haproxy.router.openshift.io/timeout: 240s
  hosts:
    - name: prod-api-manager.example.com
    secretName: prod-be787dd3-api-manager

```

15. On the deployment page, install API Connect by clicking Create.

16. Wait for **all** subsystems to be created.

When the Management subsystem is installed, you might see **backup** job pods and **stanza-create** job pods in **Error** state; for example:

m1-82b290a2-postgres-stanza-create-4zcgz	0/1	Error	0	35m
m1-82b290a2-postgres-full-sch-backup-2g9hm	0/1	Error	0	20m

This is expected behavior, for the following reasons:

- The **stanza-create** job normally expects buckets or subdirectories within buckets to be empty. However, since you configured the Management subsystem with your already-populated S3 bucket (where your backups exist), the job will enter the **Error** state.
- Any scheduled or manual backups will enter the **Error** state. Although you configured the Management subsystem with your already-populated S3 bucket, the new database isn't yet configured to write backups into remote storage.
- The configurator job will fail because the CP4I credentials secret that you manually restored does not match the value in the Management database. As a result, the state of new cluster is not "Ready" and will show "6/7". This error will be resolved when you restore the Management subsystem from the backup that you prepared earlier.

The errors will not prevent a successful restore, so continue to the next step.

17. Restore the subsystems in the following sequence:

Important: Make sure that you restore using the backup (locate the one with the correct backup name and ID) that you created when you prepared each subsystem for disaster recovery.

a. [Restore the Management subsystem.](#)

When you perform a restore, you must complete the restoration of the Management subsystem first. Verify that the restoration completed successfully and that the Management subsystem is Ready. When the Management subsystem is healthy, proceed to the next step and restore the Portal subsystem.

b. [Restore the Developer Portal subsystem.](#)

Use the restore type **a11** to ensure that you restore the complete subsystem and all Portal sites.

c. [Restore the Analytics subsystem.](#)

18. Force the configurator to run again.

The cluster will still not be ready after restore at this stage because the configurator has yet to successfully complete (refer to point 12). In order for the configurator to run again, delete the associated job so that a new pod will start running:

a. Run the following command to get the list of jobs:

```
oc get jobs -n <APIC_namespace>
```

b. Run the following command to determine the name of your API Connect instance:

```
oc get apiconnectcluster -n <APIC_namespace>
```

c. Run the following command to delete the configurator job:

```
oc -n <APIC_namespace> delete job <instance_name>-configurator
```

19. Version 10.0.2 or earlier: Update the Management OIDC credentials as explained in [Configuring the OIDC credentials on Cloud Pak for Integration](#).

20. Verify that the recovery was successful:

- Ensure that you can log in to the Cloud Manager UI.
- Verify that your provider organizations exist.
- Ensure that you can log in to each Developer portal.
- Ensure that the Analytics dashboard contains all of the analytics data that you preserved.

21. After the successful recovery, remove the annotation that you added to the **apiconnectcluster** CR in step [13](#).

If you skipped step [13](#), then skip this step as well.

- Using the Platform UI:

- Edit the API Management instance and click the YAML tab to edit the CR.
- Delete the following statement from the `spec.metadata.annotations` section:

```
apiconnect-operator/deployment-mode: disasterRecovery
```

- Using the CLI:
 - Get the name of the CR by running the following command:

```
oc get apiconnectcluster -o name -n <APIC_namespace>
```

The response looks like the following example:

```
apiconnectcluster.apiconnect.domain/instance_name
```

- Remove the annotation by running the following command, making sure to include the trailing "-" on `deployment-mode-` to indicate the removal:

```
oc -n <APIC_namespace> annotate apiconnectcluster/instance_name apiconnect-operator/deployment-mode-
```

The response looks like the following example:

```
apiconnectcluster.apiconnect.domain/instance_name annotated
```

- [Configuring the OIDC credentials on Cloud Pak for Integration](#)

When you recover API Connect V10.0.x, you must manually update the values for the Management OIDC credentials (`client_id` and `client_secret`) in the Common Services User Registry.

Configuring the OIDC credentials on Cloud Pak for Integration

When you recover API Connect V10.0.x, you must manually update the values for the Management OIDC credentials (`client_id` and `client_secret`) in the Common Services User Registry.

Before you begin

This task is required for all of the 10.0.x releases up to, and including, version 10.0.2 (skip this task for version 10.0.3 and later). The procedure requires you to log in to the Cloud Manager on the new API Connect deployment. Use the Cloud Manager administrator username and password that you saved while preparing for disaster recovery.

About this task

When you deploy API Connect, the OIDC `client_id` and `client_secret` resources are automatically generated and added to the Common Services User Registry. These resources enable single sign-on with the IBM Cloud Pak platform. When you reinstall API Connect as part of disaster recovery, the original versions of those resources are retained but cannot be used. You must update the Common Services User Registry and replace the original values with the values from the new deployment.

Procedure

1. Retrieve the `client_id` and `client_secret` from the stored secret in API Connect.

- a. Run the following command to retrieve the `client_id`:

```
oc -n <APIC_namespace> get secret <instance_name>-oidc-client -o jsonpath="{.data.CLIENT_ID}" | base64 -D
```

- b. Run the following command to retrieve the `client_secret`:

```
oc -n <APIC_namespace> get secret <instance_name>-oidc-client -o jsonpath="{.data.CLIENT_SECRET}" | base64 -D
```

2. Update the Common Services User Registry with the new values.

Using the Cloud Manager UI:

- a. In API Connect, open the Cloud Manager interface.
- b. Select the Cloud Manager User Registry.
You will use the Cloud Manager User Registry to modify settings for the Common Service User Registry.
- c. Log in to the Cloud Manager User Registry using the administrator username and password that you saved while preparing for disaster recovery.
- d. In Cloud Manager, click Resources, > User Registries and edit Common Services User Registry.
- e. In the Client information section, paste the `client_id` value from step 1 into the Client ID field.
- f. Paste the `client_secret` value from step 1 into the Client secret field.
- g. Save your changes.

Using the toolkit CLI:

- a. Run the following command to determine the `<mgmt_endpoint_url>` that you will use for accessing the management server:

```
oc -n <APIC_namespace> get mgmt <instance_name> -o jsonpath="{.status.zenRoute}" && echo ""
```

- b. Log in to the management server:

- i. Run the following command to start the login process:

```
apic login --server <mgmt_endpoint_url>
```

- ii. Provide your login credentials:

- **Realm?** Type your realm (for example: `admin/default-idp-1`) and press Enter.
The realm refers to your identity provider. For information on determining your realm value, see [How to determine the identity provider](#).
- **Username?** Type your Cloud Manager administrator username and press Enter.

- **Password?** Type the password for your Cloud Manager administrator account and press Enter.
- c. Run the following command to download the Common Service User Registry as a YAML file:

```
apic user-registries:get common-services --server <mgmt_endpoint_URL> -o admin --fields name,configuration
```

- d. Edit the file and update the `client_ID` and `client_secret` settings with the values you obtained in step 1.
 e. Save and close the file.
 f. Run the following command to upload the modified file to Cloud Manager:

```
apic user-registries:update common-services --server <mgmt_endpoint_URL> -o admin common-services.yaml
```

Disabling and re-enabling the Analytics subsystem on Kubernetes

During maintenance of an API Connect cluster, you can shut down the Analytics subsystem by disabling the ingestion, and storage microservices so that those processes are not running and there is no way to reach them.

Before you begin

This operation requires the use of the Analytics CLI. [Using the Analytics CLI](#).

About this task

When you disable Analytics, there will be no data in either the API Manager Dashboards or third-party offloads. Disabling the Analytics subsystem stops the collection and storage of data as well as making the subsystem unavailable.

Procedure

1. Backup your Analytics data as explained in [Backing up and restoring the analytics database](#).
2. Log in to the Management server: [Toolkit CLI Login](#).

```
apic login --server <management server platform api endpoint> --username <username> --realm <realm>
```

Where:

- `<username>` is the cloud admin user.
- `<realm>` is the user registry realm for the specified `<username>`. By default this is `admin/default-idp-1`.

3. Disable shard allocation for storage by following these steps:

Disabling shard allocation is optional, but is recommended because it prevents new shards from being created for replication when a node is unavailable, and helps avoid corruption issues in full system restarts.

- a. Create a json file called `analytics_persistent_settings.json` and insert the contents:

```
{"persistent":{"cluster.routing.allocation.enable":"none"}}
```

- b. Run the following Analytics CLI command:

```
apic --mode analytics clustermgmt:putSettings --server <management server platform api endpoint> --analytics-service <analytics service name> --format json analytics_persistent_settings.json
```

where:

- `<management server platform api endpoint>` is the Management server platform api endpoint.
- `<analytics service name>` is the name of Analytics service as configured in the Cloud Manager.

The output from this command should be:

```
{"acknowledged":true,"persistent":{"cluster":{"routing":{"allocation":{"enable":"none"}}},"transient":{}}
```

4. Perform a synced flush of storage by running the following command:

Flushing the storage is optional, but is recommended because it helps to avoid losing the data that was not yet written to disk before you restart pods. This step flushes all current index operations synchronously and attempts to write everything that is in flux to disk.

```
apic --mode analytics clustermgmt:postFlush --server <management server platform api endpoint> --analytics-service <analytics service name> --format json
```

where:



- `<management server platform api endpoint>` is the Management server platform endpoint.
- `<analytics service name>` is the name of Analytics service as configured in the Cloud Manager.

When you flush storage, the response looks like the following example, with the `failed` count set to zero:

```
{
  "_shards": {
    "total": 16,
    "successful": 6,
    "failed": 0
  }
}
```


This operation occasionally fails, but it is safe to rerun it multiple times until it passes and the "failed" count is zero.

5. Unassociate the Analytics service from all gateway services.

- a. In Cloud Manager, click  Topology.
- b. In the section for the Availability Zone that contains the Analytics service, locate the Gateway service that the Analytics service is associated with.
- c. Click  and select Unassociate.

For more information, see [Associating an analytics service with a gateway service](#).

6. Unregister the Analytics service from Cloud Manager:

- a. In Cloud Manager, click  Topology.
- b. In the section for the Availability Zone that contains the Analytics service, locate the Analytics service and click Delete.

For more information, see [Registering an analytics service](#).

7. Set replicas to zero for the analytics pods by editing the deployments and statefulsets that manage them: `director`, `mtls-gw`, `ingestion`, and `storage`:

```
kubectl edit deploy analytics-director -n <namespace>
kubectl edit deploy analytics-mtls-gw
kubectl edit sts analytics-ingestion
kubectl edit sts analytics-storage
```

setting `replicas` to zero:

```
spec:
  ...
  replicas: 0
  ...
```

The analytics `director`, `mtls-gw`, `ingestion`, and `storage` pods should all terminate.

Re-enabling the analytics subsystem

8. Repeat step 7 but in reverse order and setting the `replicas` value back to what it was before.

9. Wait for all the pods to be running:

```
kubectl get deploy -n <namespace> | grep analytics
analytics-director          1/1      1          1          7h10m
analytics-mtls-gw          1/1      1          1          7h6m

kubectl get sts -n <namespace> | grep analytics
analytics-ingestion        1/1      7h10m
analytics-storage         1/1      7h12m
```

10. Re-register the analytics service: [Registering an analytics service](#)

11. Re-associate your analytics service with your gateways [Associate new analytics service\(s\) with gateway\(s\)](#)

12. Enable shard allocation by following these steps:

Note: If you did not disable shard allocation, skip this step.

a. Create a json file called `analytics_persistent_settings.json` and insert the contents:

```
{"persistent": {"cluster.routing.allocation.enable": "all"}}
```

b. Run the following Analytics CLI command:

```
apic --mode analytics clustermgmt:putSettings --server <management server platform api endpoint> --analytics-service
<analytics service name> --format json analytics_persistent_settings.json
```

where:

- `<management server platform api endpoint>` is the Management server platform api endpoint.
- `<analytics service name>` is the name of Analytics service as configured in the Cloud Manager.

When you enable shard allocation, the response looks like the following example:

```
{"acknowledged": true, "persistent": {"cluster": {"routing": {"allocation": {"enable": "all"}}}, "transient": {}}
```

Disabling and re-enabling the Analytics subsystem on OpenShift and Cloud Pak for Integration

During maintenance of an API Connect cluster, you can shut down the Analytics subsystem by disabling the client, ingestion, and storage microservices so that those processes are not running and there is no way to reach them.

Before you begin

This operation requires the use of the Analytics CLI. [Using the Analytics CLI](#).

About this task

Disabling the Analytics subsystem stops the collection and storage of data as well as making the subsystem unavailable.

When you disable Analytics, there will be no data in either the API Manager Dashboards or third-party offloads.

Procedure

1. Backup your Analytics data as explained in [Backing up and restoring the Analytics database on OpenShift and Cloud Pak for Integration](#).
2. Log in to the Management server: [Toolkit CLI Login](#).

```
apic login --server <management server platform api endpoint> --username <username> --realm <realm>
```

Where:

- `<username>` is the cloud admin user.
- `<realm>` is the user registry realm for the specified `<username>`. By default this is `admin/default-idp-1`.

3. Disable shard allocation for storage by following these steps:

Disabling shard allocation is optional, but is recommended because it prevents new shards from being created for replication when a node is unavailable, and helps avoid corruption issues in full system restarts.

- a. Create a json file called `analytics_persistent_settings.json` and insert the contents:

```
{"persistent": {"cluster.routing.allocation.enable": "none"}}
```

- b. Run the following Analytics CLI command:

```
apic --mode analytics clustermgmt:putSettings --server <management server platform api endpoint> --analytics-service <analytics service name> --format json analytics_persistent_settings.json
```

where:

- `<management server platform api endpoint>` is the Management server platform api endpoint.
- `<analytics service name>` is the name of Analytics service as configured in the Cloud Manager.

The output from this command should be:

```
{"acknowledged": true, "persistent": {"cluster": {"routing": {"allocation": {"enable": "none"}}}}, "transient": {}}
```

4. Perform a synced flush of storage by running the following command:

Flushing the storage is optional, but is recommended because it helps to avoid losing the data that was not yet written to disk before you restart pods. This step flushes all current index operations synchronously and attempts to write everything that is in flux to disk.

```
apic --mode analytics clustermgmt:postFlush --server <management server platform api endpoint> --analytics-service <analytics service name> --format json
```

where:



- `<management server platform api endpoint>` is the Management server platform endpoint.
- `<analytics service name>` is the name of Analytics service as configured in the Cloud Manager.

When you flush storage, the response looks like the following example, with the *failed* count set to zero:

```
{
  "_shards": {
    "total": 16,
    "successful": 6,
    "failed": 0
  }
}
```


This operation occasionally fails, but it is safe to rerun it multiple times until it passes and the "failed" count is zero.

5. Unassociate the Analytics service from all gateway services.

- a. In Cloud Manager, click  Topology.
- b. In the section for the Availability Zone that contains the Analytics service, locate the Gateway service that the Analytics service is associated with.
- c. Click  and select Unassociate.

For more information, see [Associating an analytics service with a gateway service](#).

6. Unregister the Analytics service from Cloud Manager:

- a. In Cloud Manager, click  Topology.
- b. In the section for the Availability Zone that contains the Analytics service, locate the Analytics service and click Delete.

For more information, see [Registering an analytics service](#).

7. Set replicas to zero for the analytics pods by editing the deployments and statefulsets that manage them: `director`, `mtls-gw`, `ingestion`, and `storage`:

```
oc edit deploy analytics-director -n <namespace>
oc edit deploy analytics-mtls-gw
oc edit sts analytics-ingestion
oc edit sts analytics-storage
```

setting *replicas* to zero:

```
spec:
  ...
  replicas: 0
  ...
```

The analytics `director`, `mtls-gw`, `ingestion`, and `storage` pods should all terminate.

Re-enabling the analytics subsystem

8. Repeat step 7 but in reverse order and setting the *replicas* value back to what it was before.
9. Wait for all the pods to be running:

```
oc get deploy -n <namespace> | grep analytics
analytics-director      1/1      1      1      7h10m
analytics-mtls-gw      1/1      1      1      7h6m

oc get sts -n <namespace> | grep analytics
analytics-ingestion    1/1      7h10m
analytics-storage     1/1      7h12m
```

10. Re-register the analytics service: [Registering an analytics service](#)
11. Re-associate your analytics service with your gateways [Associate new analytics service\(s\) with gateway\(s\)](#)
12. Enable shard allocation by following these steps:

Note: If you did not disable shard allocation, skip this step.

- a. Create a json file called `analytics_persistent_settings.json` and insert the contents:

```
{"persistent": {"cluster.routing.allocation.enable": "all"}}
```

- b. Run the following Analytics CLI command:

```
apic --mode analytics clustermgmt:putSettings --server <management server platform api endpoint> --analytics-service <analytics service name> --format json analytics_persistent_settings.json
```

where:

- `<management server platform api endpoint>` is the Management server platform api endpoint.
- `<analytics service name>` is the name of Analytics service as configured in the Cloud Manager.

When you enable shard allocation, the response looks like the following example:

```
{"acknowledged": true, "persistent": {"cluster": {"routing": {"allocation": {"enable": "all"}}}, "transient": {}}
```

Maintaining a two data center deployment

How to maintain a two data center disaster recovery (DR) deployment on Kubernetes and OpenShift, including information about normal operation, failure handling, and recovery.

See the following topics for information about failure handling, backup, upgrade, and removing a 2DCDR deployment.

For general information about two data center disaster recovery in API Connect, see [Two data center deployment strategy on Kubernetes and OpenShift](#).

- [Key concepts of 2DCDR and failure scenarios](#)
Understand how API Connect 2DCDR works and what to do when a data center fails.
- [How to failover API Connect from the active to the warm-standby data center](#)
How to complete service failover in a 2DCDR deployment on Kubernetes, OpenShift, and Cloud Pak for Integration.
- [Recovering from a failover of a two data center deployment](#)
What to do after completing a 2DCDR failover.
- [Backup and restore requirements for a two data center deployment](#)
Points to consider when you configure backup and restore settings on a two data center disaster recovery (2DCDR) deployment.
- [Removing a two data center deployment](#)
Convert your 2DCDR deployment to a stand-alone deployment on Kubernetes, OpenShift, and Cloud Pak for Integration.
- [Verifying replication between data centers](#)
How to verify that your API Connect data is replicating to your warm-standby data center.
- [Troubleshooting a two data center deployment](#)
What to check when a 2DCDR deployment is not starting or not replicating data.

Related concepts

- [Two data center deployment strategy on Kubernetes and OpenShift](#)

Key concepts of 2DCDR and failure scenarios

Understand how API Connect 2DCDR works and what to do when a data center fails.

This topic discusses the concepts around the operation and maintenance of an existing API Connect 2DCDR deployment. For information on the planning and installation of a 2DCDR deployment, see: [Two data center deployment strategy on Kubernetes and OpenShift](#).

Failover of services and reverting to a stand-alone deployment

Failover means the API Connect management and portal subsystems on the warm-standby data center, become the active management and portal subsystems. Whether it is possible to convert the original active data center to be the warm-standby at the same time depends on the problems the active data center has that required the failover. However, the failed active data center must be set to warm-standby before you re-establish its connectivity to the new active data center.

Important: When the active data center is set to warm-standby, all data is deleted from its management database. The management data from the new active (what was the warm-standby) is replicated to the new warm-standby data center. Do not set the active data center to warm-standby until you confirm that data replication is working, see: [Verifying replication between data centers](#), or you have backups of your management data.

If you want to revert to a stand-alone API Connect deployment, before you start the process, ensure that the data center where you choose to keep API Connect is set to active.

- When API Connect is reverted to be a stand-alone deployment on the warm-standby data center, all management and portal data is deleted from that data center. It becomes an empty stand-alone deployment.
- When API Connect on the active data center is reverted to be a stand-alone deployment, all management and portal data remains.

Failure scenarios

The action that you should take when a failure occurs depends on the type of failure and which data center the failure occurs in:

API Connect failure in the active data center.

When the active data center is still working correctly, and still has connectivity to the warm-standby data center, but API Connect is not working correctly:

1. It might be possible to recover API Connect on the active data center. Review the troubleshooting documentation:
 - [Troubleshooting two data center replication problems](#).
 - [Installation troubleshooting](#).
 - [Troubleshooting operational problems](#).
2. Follow the failover steps to make the warm-standby the active data center. If your failed active data center cannot be changed to warm-standby, then disable the network link used by API Connect between the data centers.
3. Gather API Connect post-mortem logs from the failed data center and open a support request: [Gathering post-mortem logs](#).
4. Do not restore the network connection between data centers until API Connect on the failed data center is recovered and set to warm-standby.

Complete active data center failure

When the active data center has a failure, and API Connect is not accessible, follow these steps:

1. Disable the network connection between the active and the warm-standby data center.
2. Follow the failover steps to make the warm-standby the active data center: [How to failover API Connect from the active to the warm-standby data center](#).
3. Do not restore the network connection between data centers until API Connect on the failed data center is recovered and set to warm-standby.

Network failure that prevents API calls or user access to Management and Portal subsystems on active data center.

In this scenario, the API Connect 2DCDR deployment is working correctly, but unusable because the active data center is not accessible for API calls or for Management and Portal UI/CLI/REST API users.

1. Complete a failover, setting warm-standby to active, and the active to warm-standby: [How to failover API Connect from the active to the warm-standby data center](#).

API Connect failure on the Warm-standby data center

Where the API Connect pods on the warm-standby data center are not running as expected.

1. Review the troubleshooting documentation [Troubleshooting two data center replication problems](#).
2. Gather API Connect post-mortem logs from the failed data center and open a support case: [Gathering post-mortem logs](#).

Warm-standby data center failure

The data center where API Connect was running as warm-standby fails. In this case when the data center is recovered:

1. Verify that API Connect is running correctly as warm-standby and that the data from the active is replicated to it: [Verifying replication between data centers](#).

Network failure between data centers

When the network connection between data centers is recovered, verify that 2DCDR database replication resumes: [Verifying replication between data centers](#).

Operational states

API Connect has various transitory states that it passes through during startup and failover. The current 2DCDR operational state can be observed from the CR status section, for example:

```
kubectl describe mgmt -n <namespace>
...
Status:
...
Ha Mode: active
```

Table 1. 2DCDR Management subsystem operational states

Operational state	Description
progressing to active	Pods are progressing to the active state, but the subsystem is not capable of serving traffic.
active	All of the pods are ready and in the correct disaster recovery state for the active data center.
progressing to passive	Pods are progressing to the warm-standby state. The subsystem is not capable of serving traffic.
passive	All of the pods are ready and in the correct disaster recovery state for the warm-standby data center.

Table 2. 2DCDR Portal subsystem operational states

Operational state	Description
progressing to active	Pods are progressing to the active state, but the subsystem is not capable of serving traffic.
progressing to active (ready for traffic)	At least one pod of each type is ready for traffic. The dynamic router can be linked to this service.
active	All of the pods are ready and in the correct disaster recovery state for the active data center.
progressing to passive	Pods are progressing to the warm-standby state. The subsystem is not capable of serving traffic.
progressing to passive (ready for traffic)	At least one pod of each type is ready for traffic.
passive	All of the pods are ready and in the correct disaster recovery state for the warm-standby data center.
progressing to down	Portal is preparing to do a factory reset. This is what happens to a warm-standby portal data center when the <code>multiSiteHA</code> section is removed.

How to failover API Connect from the active to the warm-standby data center

How to complete service failover in a 2DCDR deployment on Kubernetes, OpenShift, and Cloud Pak for Integration.

Before you begin

Ensure that you have read and understand the 2DCDR concepts and reviewed the failure scenarios that are described in [Key concepts of 2DCDR and failure scenarios](#). Do not proceed with the failover until you confirm that it is the correct course of action for your situation.

About this task

Carefully follow the steps in this topic. Be aware of the following important points:

- The first step in the process is to set the active data center to warm-standby. When the active data center is set to warm-standby, all data is deleted from the active data center's management database, to be replaced by data copied from the warm-standby when it becomes active. Do not proceed with failover if you suspect there is also a problem on the warm-standby data center and you are unsure it has the most recent management data. See [Verifying replication between data centers](#), and consider restoring your active site from backup instead of attempting a failover: [Backup and restore requirements for a two data center deployment](#).

- If the warm-standby data center is offline for more than 24 hours, there can be issues with the disk space on the active data center. In this case, you should revert your deployment to a single data center topology. For more information, see [Removing a two data center deployment](#).
- Avoid both data centers being set to active at the same time while the network links between data centers are enabled. This causes split-brain.
- If the active data center failure prevents you from updating the active data center API Connect custom resources (CRs), then you should disable the network links used by API Connect between the data centers.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place. If you are using a top-level CR you must edit the `multiSiteHA` section for the subsystem in the top-level CR, instead of directly in the subsystem CRs. If the subsystem section is not included in the top-level CR, copy and paste the section from the subsystem CR to the top-level CR.

Procedure

- **Management subsystem failover. In this example, DC1 is the active and DC2 is the warm-standby.**

To initiate a failover from DC1 to DC2, you first must set DC1 to warm-standby before you set the DC2 data center to active. This action is needed to prevent split-brain, as there can't be two active data centers at the same time. The following instructions show how to failover DC1 to DC2 for the Management subsystem.

1. Set the DC1 Management subsystem to be warm-standby.

Edit the DC1 `ManagementCluster` custom resource (CR) by running the following command:

```
kubectl edit mgmt -n <namespace>
```

and change the `mode` to `passive`:

```
multiSiteHA:
  mode: passive
```

Note: If API Connect on DC1 is down, such that you can't set it to be warm-standby, you must ensure that the network links between DC1 and DC2 are disabled before you set DC2 to be active. You must then not restore the network links until you can set DC1 to be warm-standby.

2. Run the following command on DC1 to check that the Management subsystem HA mode is no longer `active`:

```
kubectl describe mgmt -n <namespace>
```

During transition to warm-standby, the `Ha mode` part of the `Status` output shows `progressing to passive`. When the transition is complete, it shows `passive`.

```
Status:
...
  Ha mode:                passive
...
```

You can proceed to set DC2 to active while DC1 is still `progressing to passive`.

3. Change the DC2 Management subsystem from warm-standby to active.

Edit the DC2 `ManagementCluster` custom resource (CR) by running the following command:

```
kubectl edit mgmt -n <namespace>
```

and change the `mode` to `active`:

```
multiSiteHA:
  mode: active
```

4. Run the following command on DC2 to check that the Management subsystem is ready for traffic:

```
kubectl describe mgmt -n <namespace>
```

The services are ready for traffic when the `Ha mode` part of the `Status` object is set on DC2 to `active`. For example:

```
Status:
...
  Ha mode:                active
...
```

Note: Failover can take 10 minutes or more to complete. If there are any problems, you can check the API Connect operator pod log for errors (search for the word "Multi").

5. Update your dynamic router to redirect all traffic to DC2 instead of DC1. Until the DC2 site becomes `active`, the UIs might not be operational.

- **Developer Portal subsystem failover**

If you have multiple portal services, you can failover a specific portal service, or all of the portal services that are running in a particular data center.

The following instructions show how to failover DC1 to DC2 for the Developer Portal subsystem. If you have multiple Developer Portal services, you must repeat these steps for each Developer Portal subsystem that you want to failover.

Note:

- Developer Portal failover does result in a temporary Developer Portal website outage until the warm-standby data center is ready to accept traffic.
- If you want to monitor the failover process, you can run the following command to check the status:

```
kubectl describe ptl
```

- Do not edit the portal CR file during the failover process.

1. Set the DC1 Developer Portal subsystem to be warm-standby.

Edit the DC1 `PortalCluster` custom resource (CR) by running the following command:

```
kubectl edit ptl <ptl-cluster-name> -n <namespace>
```

and change the `mode` to `passive`:

```
multiSiteHA:
  mode: passive
```

Where `<ptl-cluster-name>` is the name of your portal cluster if you have more than one. To see a list of your portal clusters, run:

```
kubectl get PortalCluster -n <namespace>
```

Note: If API Connect on DC1 is down, such that you can't set it to be warm-standby, you must ensure that the network links between DC1 and DC2 are disabled before you set DC2 to be active. You must then not restore the network links until you can set DC1 to be warm-standby.

2. Run the following command on DC1 to check the Developer Portal subsystem status:

```
kubectl describe ptl <ptl-cluster-name> -n <namespace>
```

You can continue to the next step when the `Ha mode` part of the `Status` object is set to `progressing to passive`, or any of the `passive` states. For example:

```
Status:
...
Ha mode:           progressing to passive
...
```

Important: You must not set DC2 to active until DC1 is either in `passive` or `progressing to passive` state. In other words, DC1 and DC2 must never both be in active state at the same time.

3. Change the DC2 Developer Portal subsystem from warm-standby to active.

Edit the DC2 `PortalCluster` custom resource (CR) by running the following command:

```
kubectl edit ptl <ptl-cluster-name> -n <namespace>
```

and change the `mode` to `active`:

```
multiSiteHA:
  mode: active
```

Where `<ptl-cluster-name>` is the name of your portal cluster if you have more than one. To see a list of your portal clusters, run:

```
kubectl get PortalCluster -n <namespace>
```

4. Update your dynamic router to redirect all traffic to DC2 instead of DC1.
5. Run the following command to check that the Developer Portal services on DC1 and DC2 are ready for traffic:

```
kubectl describe ptl <ptl-cluster-name> -n <namespace>
```

The services are ready for traffic when the `Ha mode` part of the `Status` object is set on DC1 to `passive`, and is set on DC2 to `active`. For example, on DC2:

```
Status:
...
Ha mode:           active
...
```

- **Failover of both Management and Portal subsystems**

Failover the Management subsystem first, followed by the Portal subsystem.

How long it takes to complete the failover varies, and depends on hardware speed, network latency, and the size of the databases. Approximate timings are:

For the Management subsystem:

- `warm-standby to active`: 5 minutes
- `active to warm-standby`: 15 minutes

For the Developer Portal:

- `warm-standby to active`: 15 - 40 minutes
- `active to warm-standby`: 10 minutes

What to do next

As soon as the failure on the data center is resolved, the failed data center should be brought back online and relinked to the currently active data center to maintain the highest availability; see [Recovering from a failover of a two data center deployment](#) for more information.

Related concepts

- [Two data center deployment strategy on Kubernetes and OpenShift](#)
- [Recovering from a failover of a two data center deployment](#)

Related tasks

- [Installing a two data center deployment on Kubernetes](#)
- [Installing the Management subsystem cluster](#)
- [Installing the Developer Portal subsystem](#)

Recovering from a failover of a two data center deployment

What to do after completing a 2DCDR failover.

Ensure that you have read and understand the concepts of 2DCDR. See [Two data center deployment strategy on Kubernetes and OpenShift](#) and [Key concepts of 2DCDR and failure scenarios](#).

This topic describes what to do after you have completed a failover of your 2DCDR deployment, as described in [How to failover API Connect from the active to the warm-standby data center](#).

If after the failover operation, the failed data center was successfully updated to warm-standby, then verify that replication is working: [Verifying replication between data centers](#). If replication is working, you can either:

- Revert your 2DCDR deployment to the original active and warm-standby data center designations. To revert your deployment, follow the same failover steps: [How to failover API Connect from the active to the warm-standby data center](#).
- Do nothing, and continue with your current active and warm-standby data center designations.

If your failed data center could not be updated to warm-standby, then ensure that the network links between the data centers are disabled. If the network links remain enabled then a split-brain could occur if your failed data center recovers before you are able to set it to warm-standby.

When you are able to recover the failed data center, ensure that API Connect is set to warm-standby before restoring the network connectivity to the active data center.

If you expect your failed data center to be down for a long time, then convert your active data center to a stand-alone deployment:

- Remove the `multiSiteHA` section from the `spec` sections of your management and portal CRs.

The steps to restore 2DCDR when the failed data center is recovered depend on the state of API Connect on the recovered data center.

- If API Connect is still working on the recovered data center, re-enable 2DCDR as follows:
 1. Add the `multiSiteHA` section back to your working stand-alone deployment. Ensure that it is set to **active**.
 2. Ensure that the `multiSiteHA` section is set to warm-standby (**passive**) on the recovered data center.
 3. Restore the network links between data centers.
- If the API Connect installation on the failed data center is not recoverable, then reinstall API Connect on this data center, and re-enable 2DCDR as follows:
 1. Add the `multiSiteHA` section back to your working stand-alone deployment. Ensure that it is set to **active**.
 2. Ensure that the `multiSiteHA` section is set to warm-standby (**passive**) on the recovered data center.
 3. Copy the `ingress-ca X.509` certificate from the active data center and apply it on your reinstalled warm-standby data center. For more information, see [Check the ingress-ca X.509 certificates match](#).
 4. Restore the network links between data centers.

Note: If your original active data center was in a failed state for some time, when it is recovered to warm-standby state it will take time for the data from your active data center to replicate across. The time that is taken depends on the size of your management and portal databases and how many changes were made to them while your original active data center was in a failed state.

Related concepts

- [Two data center deployment strategy on Kubernetes and OpenShift](#)

Related tasks

- [How to failover API Connect from the active to the warm-standby data center](#)
- [Installing a two data center deployment on Kubernetes](#)

Backup and restore requirements for a two data center deployment

Points to consider when you configure backup and restore settings on a two data center disaster recovery (2DCDR) deployment.

The 2DCDR backup requirements are different for the Portal and Management subsystems.

Ensure that you have read and understand the concepts of 2DCDR. See [Two data center deployment strategy on Kubernetes and OpenShift](#) and [Key concepts of 2DCDR and failure scenarios](#)

Management subsystem 2DCDR backup and restore requirements

- Different remote backup locations must be specified for each data center.
- Both active and warm-standby Management subsystems run the scheduled backup job.
- To restore from a management backup, follow these steps:
 1. Make each management data center a stand-alone API Connect deployment: Remove the `multiSiteHA` section from the `spec` section of the management CRs on both data centers. Keep a copy of the `multiSiteHA` sections that you remove, they are added back at the end of the restore process.
 2. You now have two stand-alone API Connect deployments. Decide which data center is to be the active, and restore the backup from that data center. Do not attempt to restore the backup from one data center, to the other data center.
 3. Add the `multiSiteHA` section back to the management CR in the restored data center. Ensure that the `multiSiteHA` section defines the data center as the active.
 4. Add the `multiSiteHA` section back to the management CR in the other data center. Ensure that the `multiSiteHA` section defines the data center as the warm-standby (passive).
 5. The management database takes some time to resync across the data centers. To confirm that replication is working, see: [Verifying replication between data centers](#).

Portal subsystem 2DCDR backup requirements

- The backup settings for both the active and warm-standby portal must be identical.
- Only the active data center portal takes backups. On failover, the backup job of the new warm-standby is disabled, and the backup job of the new active is enabled.
- Portal restores are only done on the active data center.

Configuring backups for 2DCDR deployments

The steps for configuring backups for both subsystems are the same as for a single API Connect deployment, but keep the additional 2DCDR requirements in mind when following them:

- [Backups on Kubernetes](#)
- [Backups on OpenShift](#)

Related concepts

- [Two data center deployment strategy on Kubernetes and OpenShift](#)

Removing a two data center deployment

Convert your 2DCDR deployment to a stand-alone deployment on Kubernetes, OpenShift, and Cloud Pak for Integration.

Ensure that you have read and understand the concepts of 2DCDR. See [Two data center deployment strategy on Kubernetes and OpenShift](#) and [Key concepts of 2DCDR and failure scenarios](#).

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place. If you are using a top-level CR you must edit the `multiSiteHA` section for the subsystem in the top-level CR, instead of directly in the subsystem CRs. If the subsystem section is not included in the top-level CR, copy and paste the section from the subsystem CR to the top-level CR.

If you want to revert to a single data center topology:

1. Decide which data center you want to keep. If this data center is not the current active data center, then complete the failover operation to make it the active: [How to failover API Connect from the active to the warm-standby data center](#).

2. Remove the `spec.multiSiteHA` section from the management subsystem CR on the active data center.

a. Run:

```
kubectl edit mgmt -n <namespace>
```

b. Delete the `spec.multiSiteHA` section.

3. Remove the `spec.multiSiteHA` section from the portal subsystem CR on the active data center.

a. Run:

```
kubectl edit ptl <ptl-cluster-name> -n <namespace>
```

Where `<ptl-cluster-name>` is the name of your portal cluster if you have more than one. To see a list of your portal clusters, run:

```
kubectl get PortalCluster -n <namespace>
```

b. Delete the `spec.multiSiteHA` section.

c. Repeat for each portal cluster if you have more than one.

4. If you want to permanently revert to a single data center topology, then uninstall API Connect on the warm-standby data center. See: [Uninstalling API Connect](#). If you plan to restore your 2DCDR deployment in the future with the same warm-standby data center, then you do not need to uninstall API Connect on the warm-standby. Instead, remove the `spec.multiSiteHA` sections from the management and portal CRs, as you did on the active data center. When the `spec.multiSiteHA` section is removed from the CR of the warm-standby data center, all API Connect data is deleted from it, and it remains in an unconfigured state.

Related concepts

- [Maintaining a two data center deployment](#)
- [Backup and restore requirements for a two data center deployment](#)
- [Recovering from a failover of a two data center deployment](#)

Related tasks

- [How to failover API Connect from the active to the warm-standby data center](#)

Verifying replication between data centers

How to verify that your API Connect data is replicating to your warm-standby data center.

The Management and Portal UIs are not accessible on the warm-standby data center, so it is not possible to check that data replication is working by logging in to the UIs. Instead, you can verify that the warm-standby Management and Portal databases are in sync with command-line operations.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.

Verifying Management replication

The management CR reports a warning if it detects a problem with management data replication:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	MESSAGE
management	n/n	Warning	10.0.5.3-0	10.0.5.3-0		Management is ready. HA Status Warning - see HAStatus in CR
for details	8m59s					

```
status:
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication not working",
    "reason": "na",
    "status": "True",
    "type": "Warning"
  }
}
```

If the warning persists, then see [Troubleshooting a two data center deployment](#).

To verify replication by creating new data on the active and checking that it is copied to the warm-standby, follow these steps:

1. Login to the Cloud Manager UI.
2. Create a local user registry called `2dcdrtest`: [Configuring a Local User Registry](#).
3. Run the following command in your warm-standby data center to query the management database for the new user registry:

```
kubectl exec -n <namespace> `kubectl -n <namespace> get po -l role=master -o name` -- bash -c "psql apim -c \"select name,created_at from user_registry where name = '2dcdrtest'\""
```

Expected output is:

```
Defaulted container "database" out of: database, set-libpq-certs (init)
  name | created_at
-----+-----
 2dcdrtest | <date and time when step 2 was done>
(1 row)
```

If `2dcdrtest` is returned and has a creation time consistent with when you did step 2, then replication is working. If it is not returned, then either replication is not working, or the data is not yet replicated. If an older creation date is returned, then it's possible the registry was not deleted after a previous check, in which case delete the registry and repeat this test. If replication is not working, follow the 2DCDR troubleshooting steps: [Troubleshooting a two data center deployment](#).

4. Return to the Cloud Manager UI, and delete the `2dcdrtest` user registry.

Verifying Portal replication

Follow these steps to verify that the portal database is replicating from the active data center to the warm-standby data center.

1. In the active data center, identify the active db pod:

```
kubectl get pods -n <namespace> | grep "<active site name>-db"
```

Example output:

```
<portal instance name>-<active site name>-db-0          2/2    Running    0
22h
```

where `<active site name>` is the `siteName` property from the portal CR on your active data center. The active data center db pods have 2 containers, so show a status of 2/2. A three replica deployment has three db pods and you can pick any one of them for the next step.

2. In the active data center, run the following command:

```
kubectl exec -ti -n <namespace> <active portal db pod> -c db -- bash -ic "dbstatus 2>&1 | grep Last | sed 's/. *committed: \([0-9]*\) .*/\1/'"
```

Example output:

```
355175
```

where `<active portal db pod>` is the pod name that is returned from step 1. The command returns a number.

3. In the warm-standby data center, identify the warm-standby db pod:

```
kubectl get pods -n <namespace> | grep "<warm-standby site name>-db"
```

Example output:

```
<portal instance name>-<warm-standby site name>-db-0    2/2    Running    0
22h
```

where `<warm-standby site name>` is the `siteName` property from the portal CR on your warm-standby data center. The warm-standby data center db pods have 1 container, so show a status of 1/1. A three replica deployment has three db pods and you can pick any one of them for the next step.

4. In the warm-standby data center, run the following command:

```
kubectl exec -ti -n <namespace> <passive portal db pod> -c db -- bash -ic "dbstatus 2>&1 | grep Last | sed 's/. *committed: \([0-9]*\) .*/\1/'"
```

Example output:

```
355495
```

where `<passive portal db pod>` is the pod name that is returned from step 3. The command returns a number.

5. The numbers that are returned from steps 2 and 4 represent the portal database's last committed transaction number. When the active data center commits a new transaction the number is incremented, this update is replicated to the warm-standby data center. If replication is working the number that is returned from the warm-standby data center should always be close to the number that is returned from the active data center. It is unlikely to be the same number that is returned, unless you are able run step 4 and step 2 at precisely the same time.
6. Repeat steps 2 and 4 several times to confirm that the warm-standby number is keeping up with the active number. If the number returned from the warm-standby data center is far behind, or ahead of the number that is returned from the active, then there is likely to be a replication problem, see [Troubleshooting a two data center deployment](#).

Troubleshooting a two data center deployment

What to check when a 2DCDR deployment is not starting or not replicating data.

Typical causes of 2DCDR failures are network problems between data centers, TLS certificate mismatches, or environmental problems on one of the data centers.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.

Check that API Connect pods are running

Confirm that all the pods are ready and running:

- To check portal, run:

```
kubectl get ptl -n <namespace>
```

NAME	READY	STATUS	VERSION	RECONCILED VERSION	MESSAGE
AGE					
<portal instance name>	<ready pods>/<total number of pods>	Running	10.0.5.x	10.0.5.x-xxxx	Serving 1
site	14d				

- To check management, run:

```
kubectl get mgmt -n <namespace>
```

NAME	READY	STATUS	VERSION	RECONCILED VERSION	AGE
<management instance name>	<ready pods>/<total number of pods>	Running	10.0.5.x	10.0.5.x-xxxx	14d

The number of `<ready pods>` must match the `<total number of pods>`. If not all the pods are in ready state, run

```
kubectl get <mgmt or ptl> -n <namespace> -o yaml
```

and check the `status.conditions` output, for example:

```
conditions:
- lastTransitionTime: "2022-09-27T21:37:21Z",
  message: "Management installation in progress. Not all services are ready, pending services: analytics-proxy, apim, client-downloads-server, juhu, ldap, ...",
  reason: na
  status: "True"
  type: Pending
```

Also, check which pods are not running with:

```
kubectl get pods -n <namespace>
```

and review their logs:

```
kubectl logs <pod name> -n <namespace>
```

Note: The warm-standby management subsystem has fewer pods than the active management subsystem. The active and warm-standby portal subsystems have the same number of pods.

Check the multiSiteHA sections of the management and portal CRs

Run `kubectl describe` on your management and portal CRs to confirm that the `multiSiteHA` specification and status sections of the CR are as expected. For example, on the warm-standby management CR you should see:

```
kubectl describe mgmt -n <namespace>
```

```
...
spec:
  Multi Site HA:
    Mode: passive
    Replication Endpoint:
      Annotations:
        cert-manager.io/issuer: ingress-issuer
      Hosts:
        Name: mgmt-replication.apps.example.com
        Secret Name: mgmt-replication-server
    Replication Peer FQDN: mgmt-replication.apps.example.com
    Tls Client:
      Secret Name: mgmt-replication-client
...
Status
...
  Ha Mode: passive
...
  Multi Site HA:
    Remote Site Deployment Name: management
    Remote Site Name: passive
    Remote UUID: c736b702-b1ab-4fe2-b132-9c9d3b3a3bd3.9f986be1-14fc-4b3a-8eee-094217ce361e
    Replication Peer: https://mgmt-replication.apps.example.com/
```

Check that the `Hosts` and `Replication Peer` properties are set correctly in the `multiSiteHA` section, and that the hostnames resolve correctly in each data center.

Check the `ingress-ca` X.509 certificates match

Replication fails if both data centers do not have the same X.509 certificate in their `ingress-ca` secrets. Run the following command in both data centers to see the X.509 certificate, and check that the output is the same:

```
openssl x509 -noout -fingerprint -sha256 -in <(kubectl get secret ingress-ca -n <namespace> -o yaml | grep "^  tls.crt:" | awk '{print $2}' | base64 -d)
```

If you do not have the `openssl` command available, you can instead run only the `kubectl` part, which produces a larger output:

```
kubectl get secret ingress-ca -n <namespace> -o yaml | grep "^  tls.crt:" | awk '{print $2}' | base64 -d
```

if the outputs are different, follow these steps to synchronize the certificates:

1. If you installed API Connect on Kubernetes or OpenShift using individual subsystem CRs, determine which data center has the `ingress-ca` Kubernetes cert-manager certificate object:

```
kubectl get certificates -n <namespace> | grep ingress-ca
```

this is your source data center.

2. If you installed API Connect on Cloud Pak for Integration, use the current active data center as the source data center, and the warm-standby as the target data center.

3. Extract the `ingress-ca` secret from your source data center to a file called `new-ca-issuer-secret.yaml`:

```
kubectl get secret ingress-ca -o yaml -n <namespace> > new-ca-issuer-secret.yaml
```

4. Edit the `new-ca-issuer-secret.yaml` file and remove the `creationTimestamp`, `resourceVersion`, `uid`, `namespace`, and `managedFields`. Remove the labels and annotations sections completely. The resulting contents should include the `ingress-ca` X.509 certificate, and the secret name:

```
apiVersion: v1
data:
  ca.crt: <long cert string>
  tls.crt: <long cert string>
  tls.key: <long cert string>
kind: Secret
metadata:
  name: ingress-ca
  type: kubernetes.io/tls
```

5. Copy the `new-ca-issuer-secret.yaml` to the target data center.

Follow these steps to apply the extracted `ingress-ca` X.509 certificate on your target data center:

1. To apply the `new-ca-issuer-secret.yaml` file, run:

```
kubectl apply -f new-ca-issuer-secret.yaml -n <namespace>
```

2. Regenerate all `ingress-ca` end-entity certificates:

```
kubectl get secrets -n <namespace> -o custom-columns='NAME:.metadata.name,ISSUER:.metadata.annotations.cert-manager.io/issuer-name' --no-headers=true | grep ingress-issuer | awk '{ print $1 }' | xargs kubectl delete secret -n <namespace>
```

All affected pods should automatically restart. For more information on regenerating certificates, see: [Renewing certificates with cert-manager](#).

Check the operator and tunnel pod logs

The API Connect operator pod manages the 2DCDR deployment. The tunnel pods manage the communication between data centers.

Check the logs of the API Connect operator pod and search for the text "multi" to see any errors that are related to 2DCDR. For example:

```
kubectl logs <ibm-apiconnect operator pod> -n <operator namespace> | grep -i multi
```

The `<ibm-apiconnect operator pod>` has `ibm-apiconnect` in its name, and might be in a different namespace to your API Connect operand pods.

Check the logs of your API Connect `tunnel` pods. These pods always have `tunnel` in their name. For example:

```
kubectl logs portal-tunnel-0 -n <namespace> --since=10m
```

```
...
[ ws-tunnel stderr] 400 0eec87:fb8309:21aa2d 2022-12-02 12:27:33: 2022-12-02T12:27:33.786Z INFO tls incoming
request {"remote-addr": "10.254.20.144:55042", "uri": "/portal-active-db-0/3060"}
[ ws-tunnel stderr] 400 0eec87:fb8309:21aa2d 2022-12-02 12:27:33: 2022-12-02T12:27:33.786Z INFO tls connect to
upstream {"remote-addr": "10.254.20.144:55042", "uri": "/portal-active-db-0/3060"}
[ ws-tunnel stderr] 400 0eec87:fb8309:21aa2d 2022-12-02 12:27:33: 2022-12-02T12:27:33.817Z INFO tls closing
connection {"remote-addr": "10.254.20.144:55042", "uri": "/portal-active-db-0/3060"}
[ ws-tunnel stderr] 400 0eec87:fb8309:21aa2d 2022-12-02 12:27:33: 10.254.20.144 - - [02/Dec/2022:12:27:33 +0000] "GET
/portal-active-db-0/3060 HTTP/1.1" 101 0
```

Note: It is normal for the management tunnel pod to repeatedly log:

```
2022/12/02 12:29:17 http: TLS handshake error from 10.254.16.1:44812: EOF
```

This message can be filtered out with `grep`

```
-v:
```

```
kubectl logs management-tunnel-574bdcd865-48zh6 -n <namespace> | grep -v "http: TLS handshake error from"
```

Monitoring and health checks

Monitor your deployment and complete health checks on subsystems.

See:

- [Monitoring on Kubernetes](#)
Monitor the API Connect subsystems on Kubernetes.
- [Monitoring on OpenShift](#)
Monitor your API Connect deployment on OpenShift.
- [Monitoring on Cloud Pak for Integration](#)
Monitor your API Connect deployment on Cloud Pak for Integration.
- [Obtaining simple health check data of Developer Portal sites by using a REST API call](#)
Call a simple health check API from your external load balancer to dynamically determine whether a specific Developer Portal site in a cluster is working. This API call can be used by a load balancer to help determine where to route traffic.

Monitoring on Kubernetes

Monitor the API Connect subsystems on Kubernetes.

See:

- [Monitoring subsystems on Kubernetes](#)
Use `kubect1 get` to check the health of the API Connect clusters in your Kubernetes deployment.
- [Monitoring Postgres disk usage](#)
Monitor disk usage by the Postgres database in the management subsystem.
- [Monitoring platform health using Kubernetes events](#)
Monitor the health of your API Connect deployment by viewing the Kubernetes events that are generated for every status change in each subsystem CR.

Monitoring subsystems on Kubernetes

Use `kubect1 get` to check the health of the API Connect clusters in your Kubernetes deployment.

The APICoconnect operator controls the life-cycle of management subsystem microservices. When you install, you create the following custom resources (CRs) in order to install API Connect and all subsystems:

- `ManagementCluster` or `mgmt`
 - `AnalyticsCluster` or `a7s`
 - `PortalCluster` or `ptl`
 - `GatewayCluster` or `gw`
- Every CR has a `status` section. The APICoconnect operator regularly updates this section with the current status of the subsystem.

For example, use the following command to obtain the status of `ManagementCluster`:

```
kubect1 get mgmt -n <namespace>
```

Example:

```
kubect1 get mgmt
NAME   READY   STATUS    VERSION           RECONCILED VERSION   AGE
m1     16/16   Running   10.0.1-eus        10.0.1.4-ifix1-44-eus 24m
```

Output:

- **NAME**: Name of the Management Subsystem
- **READY**: Current number of management subsystem which are up, followed by the expected number of management subsystem components which should be up. For example: `16/16`.
- **STATUS**: Current status of Management Subsystem
During fresh install, upgrade or any maintenance operation, such as changing configuration for S3 backups, Management CR status can go to different states depending on the action, but ultimately it should **always** reach **Running** status.

If the status says `Warning`, then run `kubect1 describe mgmt -n <namespace>` and check for more details in the `Status` section. For example,

```
Status:
...
Conditions:
...
Message:          WAL Archiving is not working. Last successful archive time: 2022-07-26
20:37:14.456465 +0000 UTC. Last archive failed time: 2022-07-26 21:10:34.172539 +0000 UTC
Reason:           WALArchiveNotWorking
...
```

Note: If you see the `WAL Archiving is not working` warning message, then verify your S3 backup settings (bucket name, folder name, credentials, certificates), gather logs and open a support request.

- **VERSION**: Version chosen in `spec.version` in the management CR.
- **RECONCILED VERSION**: Current management subsystem version. Includes a build number.
- **AGE**: Age of the management subsystem CR.

Monitoring Postgres disk usage

Monitor disk usage by the Postgres database in the management subsystem.

Postgres Database is the core database used in Management Subsystem. It is important to monitor the Postgres disk usage.

In 10.0.3.0 and greater, the APIConnect operator tracks the current disk usage of the Postgres components, and regularly updates the `ManagementCluster` status. When one or more of the Postgres components occupy 50% of the PVC (persistent volume claim) capacity, the APIConnect operator changes the status from `Running` to `Warning`.

APIConnect operator **brings down** Postgres if the disk allocation reaches **80%**. In this case, you must contact IBM support.

Note that there are various types of storage classes which can be used to deploy Management Subsystem. For example, `local-storage` or `ceph block`.

- [Management Cluster disk stats](#)
- [Warning condition is populated at 50% usage](#)
- [Error condition is populated at 80% usage](#)

Management Cluster disk stats

The `ManagementCluster` instance has `.status.postgresDataStats` where the operator prints the current disk usage of Postgres components. For example:

```
kubectl get mgmt ml -o json | jq .status.postgresDataStats
[
  {
    "instanceName": "ml-ed60c42d-postgres",
    "podName": "ml-ed60c42d-postgres-86766f69cb-xs7t5",
    "pvcCapacity": 250181844992,
    "pvcName": "ml-ed60c42d-postgres",
    "pvcType": "PostgreSQL",
    "pvcUsed": 60895232,
    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "ml-ed60c42d-postgres-backrest-shared-repo",
    "podName": "ml-ed60c42d-postgres-backrest-shared-repo-859484b5f6-r7cv6",
    "pvcCapacity": 250181844992,
    "pvcName": "ml-ed60c42d-postgres-pgbr-repo",
    "pvcType": "pgBackRest",
    "pvcUsed": 16203776,
    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "ml-ed60c42d-postgres",
    "podName": "ml-ed60c42d-postgres-86766f69cb-xs7t5",
    "pvcCapacity": 250181844992,
    "pvcName": "ml-ed60c42d-postgres-wal",
    "pvcType": "WAL",
    "pvcUsed": 201338880,
    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "ml-ed60c42d-postgres-fqbg",
    "podName": "ml-ed60c42d-postgres-fqbg-84869d976c-49qgc",
    "pvcCapacity": 250181844992,
    "pvcName": "ml-ed60c42d-postgres-fqbg",
    "pvcType": "PostgreSQL",
    "pvcUsed": 59994112,
    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "ml-ed60c42d-postgres-fqbg",
    "podName": "ml-ed60c42d-postgres-fqbg-84869d976c-49qgc",
    "pvcCapacity": 250181844992,
    "pvcName": "ml-ed60c42d-postgres-fqbg-wal",
    "pvcType": "WAL",
    "pvcUsed": 201334784,
    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "ml-ed60c42d-postgres-cimo",
    "podName": "ml-ed60c42d-postgres-cimo-5769868b75-5z7ln",
    "pvcCapacity": 250181844992,
    "pvcName": "ml-ed60c42d-postgres-cimo",
    "pvcType": "PostgreSQL",
    "pvcUsed": 59994112,
    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "ml-ed60c42d-postgres-cimo",
    "podName": "ml-ed60c42d-postgres-cimo-5769868b75-5z7ln",
    "pvcCapacity": 250181844992,
    "pvcName": "ml-ed60c42d-postgres-cimo-wal",
    "pvcType": "WAL",
    "pvcUsed": 201334784,
    "pvcUsedPercentage": 0
  }
]
```

Warning condition is populated at 50% usage

When one of the Postgres components occupy 50% usage, the operator marks the overall status of the `ManagementCluster` to `Warning` with an appropriate warning message. The operator also updates the `postgresDataStats` section with the current data usage of the Postgres components.

Example of warning state:

```
kubect1 get mgmt m1
```

```
NAME   READY   STATUS    VERSION      RECONCILED VERSION   AGE
m1     17/17   Warning   10.0.1-eus   10.0.1.4-ifix1-44-eus 26h
```

```
kubect1 get mgmt m1 -o json | jq .status.phase
```

```
"Warning"
```

Example of list of warning conditions:

```
kubect1 get mgmt m1 -o json | jq .status.conditions
```

```
[
  {
    "lastTransitionTime": "2021-09-08T19:49:51Z",
    "message": "Current WAL disk usage of m1-ed60c42d-postgres is 51 percent. DATABASE SHUTDOWN starts at 80 percent utilization. Please contact IBM Support immediately",
    "reason": "disk_usage_more_than_50_percent",
    "status": "True",
    "type": "Warning"
  },
  {
    "lastTransitionTime": "2021-09-08T19:42:30Z",
    "message": "",
    "reason": "na",
    "status": "False",
    "type": "Ready"
  },
  {
    "lastTransitionTime": "2021-09-08T19:41:35Z",
    "message": "",
    "reason": "na",
    "status": "False",
    "type": "Pending"
  },
  {
    "lastTransitionTime": "2021-09-07T17:18:02Z",
    "message": "",
    "reason": "na",
    "status": "False",
    "type": "Error"
  },
  {
    "lastTransitionTime": "2021-09-07T17:18:02Z",
    "message": "",
    "reason": "na",
    "status": "False",
    "type": "Failed"
  }
]
```

The warning condition explains why the warning condition was set. For example:

```
Current WAL disk usage of m1-ed60c42d-postgres is 51 percent. DATABASE SHUTDOWN starts at 80 percent utilization. Please contact IBM Support immediately
```

If you encounter the warning, contact IBM support to fix the root cause.

Error condition is populated at 80% usage

An error condition is populated when the operator brings down Postgres because is using 80% of available disk space. Postgres is brought down to avoid problems, such as disk corruption, that can occur if the disk becomes completely filled.

```
- lastTransitionTime: "2021-08-28T01:42:56Z"
  message: Current WAL disk usage of nihar-stac-3955b42d-sitel-postgres is more
  than 70 percent, initiating DATABASE SHUTDOWN
  reason: disk_usage_more_than_70_percent
  status: "True"
  type: Error
```

If you encounter the error, contact IBM support to fix the root cause.

Important:

When `local-storage` is used, it uses the entire disk that is allocated to the worker node. In some cases, Kubernetes itself can face disk pressure before the operator reacts to 80% usage, and will start evicting pods. In these cases, you might need to increase the disk allocated to the worker node, in order to stabilize the worker node, by following [Recovering when disks are filled by the management database](#).

Monitoring platform health using Kubernetes events

Monitor the health of your API Connect deployment by viewing the Kubernetes events that are generated for every status change in each subsystem CR.

What are the Kubernetes events?

The CR for every subsystem of API Connect has a status condition section, which is regularly updated by the API Connect operator with the current status. Beginning in 10.0.5.3, the API Connect operator additionally generates a Kubernetes event whenever there is a change in the CR status.

Events are enabled in API Connect operator for the following resources.

- **APIConnectCluster** CR
- **ManagementCluster** CR
- **GatewayCluster** CR
- **AnalyticsCluster** CR
- **PortalCluster** CR

Two types of events are generated when the CR status changes:

- **Normal** - Indicates that the status change is not causing any issues.
- **Warning** - Indicates that there a problem that requires your attention.

You can automatically monitor the status condition in your existing alerting system, or by integrating the Kubernetes events into a third-party alerting solution. For more information, see the following blog post: [How to use Kubernetes events for effective alerting and monitoring](#).

Viewing the events

The default retention period of events in Kubernetes is 1 hour (controlled by `kube-apiserver`). To view the events, complete the following steps:

1. Run the `kubectl describe` against the resource you want to monitor; for example:

```
kubectl describe APIConnectCluster
```

2. In the response, scroll to the end to see the events information, which looks like the following example:

```
Name:          10.0.5.3-1
Reconciled:    10.0.5.3-1
Events:
  Type    Reason      Age   From              Message
  ----    -
  Normal  Running     10h   ibm-apiconnect-operator  All services ready.
  Normal  Deploying   10h   ibm-apiconnect-operator  Not all services are ready, pending services: management
```

Monitoring on OpenShift

Monitor your API Connect deployment on OpenShift.

See:

- [Monitoring the API Connect cluster on OpenShift](#)
Monitor the status of the API Connect cluster on OpenShift using either the command line or the OpenShift web console.
- [Monitoring Postgres disk usage on OpenShift](#)
In API Connect, you can monitor the disk space that is used by the Postgres database in the Management subsystem.
- [Monitoring platform health on Red Hat OpenShift using Kubernetes events](#)
Monitor the health of your API Connect deployment by viewing the Kubernetes events that are generated for every status change in each subsystem CR.

Monitoring the API Connect cluster on OpenShift

Monitor the status of the API Connect cluster on OpenShift using either the command line or the OpenShift web console.

You deploy the API Connect subsystems on OpenShift with the `APIConnect` operator and the `apiconnectcluster` CR. The CR contains a status section that is regularly updated by the operator to display the current status of the all the API Connect subsystems.

When you finish deploying API Connect, you can monitor the status of `apiconnectcluster` using the command line or the OpenShift web console.

- [Monitoring the API Connect cluster using the command line](#)
- [Monitoring the API Connect cluster using the OpenShift web console](#)

Version 10.0.3 and later: You can also monitor the disk usage of Postgres microservices. The `APIConnect` operator continuously tracks the disk usage of postgres microservices and provides useful information in the `ManagementCluster` custom resource. For information on viewing the postgres disk status, see [Monitoring Postgres disk usage on OpenShift](#).

Monitoring the API Connect cluster using the command line

Run the following command to display the status of the API Connect cluster:

```
oc get apiconnectcluster -n <APIC_namespace>
```

The response looks like the following example:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
production	7/7	Ready	10.0.1.5-eus	10.0.1.5-3323-eus		24h

The following list explains the values in the response:

- **NAME:** Name of the API Connect instance (in the example, the instance name is "production").
- **READY:** Current number of subsystems and components that are running / Expected number of subsystems and components that should be running.
- **STATUS:** Current status of API Connect.

If the status says **Warning**, then run `oc describe apiconnectcluster -n <namespace>` and check for more details in the **Status** sections. For example, for the management CR:

```
Status:
...
Conditions:
...
  Message:          WAL Archiving is not working. Last successful archive time: 2022-07-26
20:37:14.456465 +0000 UTC. Last archive failed time: 2022-07-26 21:10:34.172539 +0000 UTC
  Reason:           WALArchiveNotWorking
...

```

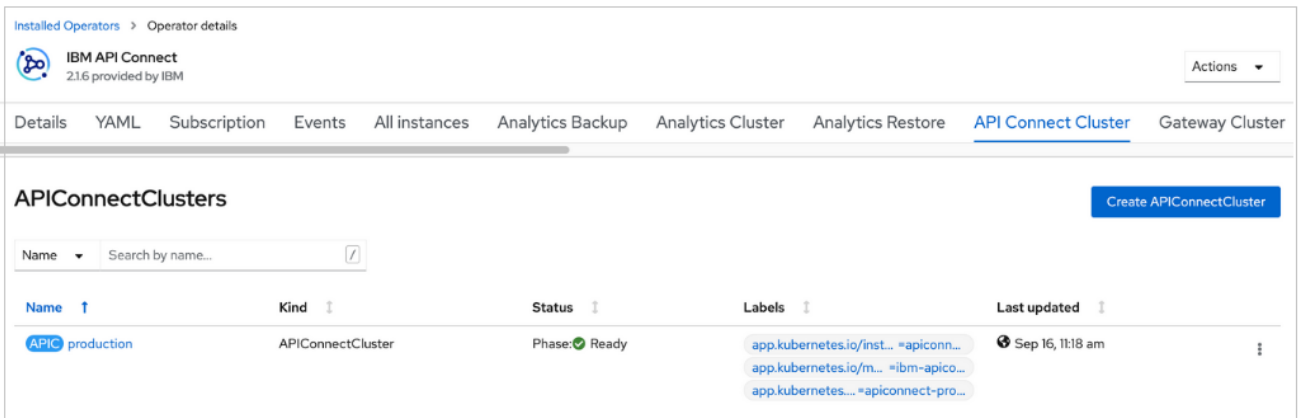
Note: If you see the **WAL Archiving is not working** warning message, then verify your S3 backup settings (bucket name, folder name, credentials, certificates), gather logs and open a support request.

- **VERSION:** API Connect version specified in the CR's `spec.version` setting.
- **RECONCILED VERSION:** Current API Connect version.
- **AGE:** Age of the `apiconnectcluster` CR.

Monitoring the API Connect cluster using the OpenShift web console

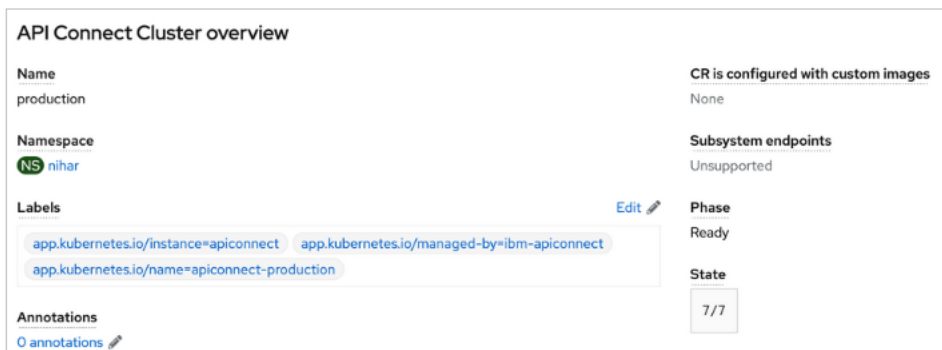
1. Start the OpenShift web console.
2. Click Installed operators.
3. Locate the namespace (project) where you installed API Connect:
 - If you installed the API Connect operator in all namespaces, select `openshift-operators`
 - if you installed the API Connect operator in a single namespace, select specific project
4. Hover next to the name of the API Connect namespace and click API Connect Cluster.
5. On the API Connect Clusters page, check the Status column.

The status should display "Ready" as in the following image:



6. Click the instance name (in the example, the instance name is production to display the status overview page.

The overview page displays high-level information about your deployment, as in the following image:



Also on the overview page, the "Conditions" section displays the list of possible service conditions, as in the following image:

Conditions				
Type	Status	Updated	Reason	Message
Warning	False	🕒 Sep 16, 11:18 am	na	-
Ready	True	🕒 Sep 16, 8:26 pm	Running	7/7
Pending	False	🕒 Sep 16, 8:26 pm	na	-
Error	False	🕒 Sep 16, 11:18 am	na	-

A healthy deployment displays "True" for the "Ready" status.

During a fresh install, an upgrade or a day-2 operation (such as a Management backup S3 configuration change), the `apiconnectcluster` CR status might change to a different state depending on the action, but ultimately it should always reach the **Ready** status.

Monitoring Postgres disk usage on OpenShift

In API Connect, you can monitor the disk space that is used by the Postgres database in the Management subsystem.

The Postgres database is the core database used in the Management subsystem. It is important to monitor the Postgres disk usage to avoid running out of space and causing an outage in your deployment.

In Version 10.0.3 and later, the **APICConnect** operator tracks the current disk usage of the Postgres components, and regularly updates the **ManagementCluster** CR's status. When one or more of the Postgres components occupy 50% of the PVC (persistent volume claim) capacity, the **APICConnect** operator changes the CR's status from **Running** to **Warning**.

Important: If the Postgres disk usage reaches 80%, then the **APICConnect** operator brings down Postgres. If this situation occurs, contact IBM Support.

Note that there are various types of storage classes which can be used to deploy Management subsystem; for example, `local-storage` or `ceph block`. When `local-storage` is used, the entire disk is allocated to the worker node. In some cases, before the **APICConnect** operator reacts to an 80% usage condition, Kubernetes itself might face disk pressure and start evicting pods. In this situation, you might want to increase the size of the disk allocated to the worker node as explained in [Recovering on OpenShift and Cloud Pak for Integration when disks are filled by the management database](#).

Viewing the current disk usage

The **ManagementCluster** instance includes the `.status.postgresDataStats` field, where the operator displays the current disk usage of Postgres components. Run the following command to get the disk usage:

```
ocp get mgmt m1 -n APIC_namespace -o json | jq .status.postgresDataStats
```

The response looks like the following example:

```
[
  {
    "instanceName": "m1-ed60c42d-postgres",
    "podName": "m1-ed60c42d-postgres-86766f69cb-xs7t5",
    "pvcCapacity": 250181844992,
    "pvcName": "m1-ed60c42d-postgres",
    "pvcType": "PostgreSQL",
    "pvcUsed": 60895232,
    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "m1-ed60c42d-postgres-backrest-shared-repo",
    "podName": "m1-ed60c42d-postgres-backrest-shared-repo-859484b5f6-r7cv6",
    "pvcCapacity": 250181844992,
    "pvcName": "m1-ed60c42d-postgres-pgbr-repo",
    "pvcType": "pgBackRest",
    "pvcUsed": 16203776,
    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "m1-ed60c42d-postgres",
    "podName": "m1-ed60c42d-postgres-86766f69cb-xs7t5",
    "pvcCapacity": 250181844992,
    "pvcName": "m1-ed60c42d-postgres-wal",
    "pvcType": "WAL",
    "pvcUsed": 201338880,
    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "m1-ed60c42d-postgres-fqbg",
    "podName": "m1-ed60c42d-postgres-fqbg-84869d976c-49qgc",
    "pvcCapacity": 250181844992,
    "pvcName": "m1-ed60c42d-postgres-fqbg",
    "pvcType": "PostgreSQL",
    "pvcUsed": 59994112,
    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "m1-ed60c42d-postgres-fqbg",
    "podName": "m1-ed60c42d-postgres-fqbg-84869d976c-49qgc",
    "pvcCapacity": 250181844992,
    "pvcName": "m1-ed60c42d-postgres-fqbg-wal",
    "pvcType": "WAL",
    "pvcUsed": 201334784,

```

```

    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "m1-ed60c42d-postgres-cimo",
    "podName": "m1-ed60c42d-postgres-cimo-5769868b75-5z7ln",
    "pvcCapacity": 250181844992,
    "pvcName": "m1-ed60c42d-postgres-cimo",
    "pvcType": "PostgreSQL",
    "pvcUsed": 59994112,
    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "m1-ed60c42d-postgres-cimo",
    "podName": "m1-ed60c42d-postgres-cimo-5769868b75-5z7ln",
    "pvcCapacity": 250181844992,
    "pvcName": "m1-ed60c42d-postgres-cimo-wal",
    "pvcType": "WAL",
    "pvcUsed": 201334784,
    "pvcUsedPercentage": 0
  }
]

```

Warning condition is populated at 50% usage

When one of the Postgres components uses 50% of its allocated space, the operator changes the overall status of the **ManagementCluster** to **Warning** with an appropriate warning message. The operator also updates the **postgresDataStats** section with the current data usage of the Postgres components.

Attention: If you encounter the **Warning** condition, contact IBM Support for help correcting the root cause.

To view the status, run the following command:

```
oc get mgmt m1
```

The following example response shows the warning state:

NAME	READY	STATUS	VERSION	RECONCILED	VERSION	AGE
m1	17/17	Warning	10.0.1-eus	10.0.1.4-ifix1-44-eus		26h

To view just the status, run the following command:

```
oc get mgmt m1 -o json | jq .status.phase
```

In this example, the status displayed in the response is "Warning":

```
"Warning"
```

To see the list of current status conditions, run the following command:

```
oc get mgmt m1 -o json | jq .status.conditions
```

The following example response displays the list of status conditions. Notice that the **Warning** condition is set to "True" and the other conditions are set to "False":

```

[
  {
    "lastTransitionTime": "2021-09-08T19:49:51Z",
    "message": "Current WAL disk usage of m1-ed60c42d-postgres is 51 percent. DATABASE SHUTDOWN starts at 80 percent utilization. Please contact IBM Support immediately",
    "reason": "disk_usage_more_than_50_percent",
    "status": "True",
    "type": "Warning"
  },
  {
    "lastTransitionTime": "2021-09-08T19:42:30Z",
    "message": "",
    "reason": "na",
    "status": "False",
    "type": "Ready"
  },
  {
    "lastTransitionTime": "2021-09-08T19:41:35Z",
    "message": "",
    "reason": "na",
    "status": "False",
    "type": "Pending"
  },
  {
    "lastTransitionTime": "2021-09-07T17:18:02Z",
    "message": "",
    "reason": "na",
    "status": "False",
    "type": "Error"
  },
  {
    "lastTransitionTime": "2021-09-07T17:18:02Z",
    "message": "",
    "reason": "na",
    "status": "False",
    "type": "Failed"
  }
]

```

The warning condition includes an explanation of why the condition was set to "True"; for example:

Current WAL disk usage of ml-ed60c42d-postgres is 51 percent. DATABASE SHUTDOWN starts at 80 percent utilization. Please contact IBM Support immediately

Error condition is populated at 80% usage

When one of the Postgres components uses 80% of its allocated space, the operator changes the overall status of the **ManagementCluster** to **Error** and brings down Postgres avoid problems that can occur if the disk becomes completely filled.

Attention: If you encounter the **Error** condition, contact IBM Support for help correcting the root cause.

In this case, when you run the following command to view the status conditions, you will see that the "Error" condition is set to "True":

```
oc get mgmt ml -o json | jq .status.conditions
```

For example, the following condition shows the **Error** status set to "True" due to disk usage:

```
- lastTransitionTime: "2021-08-28T01:42:56Z"
  message: Current WAL disk usage of nihar-stac-3955b42d-sitel-postgres is more
           than 70 percent, initiating DATABASE SHUTDOWN
  reason: disk_usage_more_than_70_percent
  status: "True"
  type: Error
```

Monitoring platform health on Red Hat OpenShift using Kubernetes events

Monitor the health of your API Connect deployment by viewing the Kubernetes events that are generated for every status change in each subsystem CR.

What are the Kubernetes events?

The CR for every subsystem of API Connect has a status condition section, which is regularly updated by the API Connect operator with the current status. Beginning in 10.0.5.3, the API Connect operator additionally generates a Kubernetes event whenever there is a change in the CR status.

Events are enabled in API Connect operator for the following resources.

- **APIConnectCluster** CR
- **ManagementCluster** CR
- **GatewayCluster** CR
- **AnalyticsCluster** CR
- **PortalCluster** CR

Two types of events are generated when the CR status changes:

- **Normal** - Indicates that the status change is not causing any issues.
- **Warning** - Indicates that there a problem that requires your attention.

You can automatically monitor the status condition in your existing alerting system, or by integrating the Kubernetes events into a third-party alerting solution. For more information, see the following blog post: [How to use Kubernetes events for effective alerting and monitoring](#).

Viewing the events

The default retention period of events in Red Hat OpenShift is 3 hours (controlled by **kube-apiserver**). To view the events, complete the following steps:

1. In the Web UI, click Operators > Installed Operators > IBM API Connect > API Connect Clusters.
2. Select an instance by clicking its name: Management cluster, Gateway cluster, Portal cluster, or Analytics cluster. If you installed using the top-level **APIConnectCluster** CR, you can additionally select API Connect cluster.
3. Select the Events tab to see all of the events that were generated for that instance.

Monitoring on Cloud Pak for Integration

Monitor your API Connect deployment on Cloud Pak for Integration.

See:

- [Monitoring the API Connect cluster on Cloud Pak for Integration](#)
Monitor the status of the API Connect cluster on Cloud Pak for Integration using either the command line or the IBM Cloud Pak Platform UI.
- [Monitoring Postgres disk usage on Cloud Pak for Integration](#)
In API Connect, you can monitor the disk space that is used by the Postgres database in the Management subsystem.

Monitoring the API Connect cluster on Cloud Pak for Integration

Monitor the status of the API Connect cluster on Cloud Pak for Integration using either the command line or the IBM Cloud Pak Platform UI.

In Cloud Pak for Integration, the API Management capability is provide by API Connect. You deploy the API Connect subsystems on Cloud Pak for Integration with the **APIConnect** operator and the **apiconnectcluster** CR. The CR contains a status section that is regularly updated by the operator to display the current status of the all the API Connect subsystems.

When you finish deploying API Connect, you can monitor the status of `apiconnectcluster` using the command line or the OpenShift web console.

- [Monitoring the API Connect cluster using the command line](#)
- [Monitoring the API Connect cluster using the Platform UI](#)

API Connect 10.0.3 and later: You can also monitor the disk usage of Postgres microservices. The `APIConnect` operator continuously tracks the disk usage of postgres microservices and provides useful information in the `ManagementCluster` custom resource. For information on viewing the postgres disk status, see [Monitoring Postgres disk usage on Cloud Pak for Integration](#).

Monitoring the API Connect cluster using the command line

Run the following command to display the status of the API Connect cluster:

```
oc get apiconnectcluster -n <APIC_namespace>
```

The response looks like the following example:

NAME	READY	STATUS	VERSION	RECONCILED VERSION	AGE
production	7/7	Ready	10.0.1.5-eus	10.0.1.5-3323-eus	24h

The following list explains the values in the response:

- **NAME:** Name of the API Connect instance (in the example, the instance name is "production").
- **READY:** Current number of subsystems and components that are running / Expected number of subsystems and components that should be running.
- **STATUS:** Current status of API Connect.
If the status says `Warning`, then run `oc describe apiconnectcluster -n <namespace>` and check for more details in the `Status` sections. For example, for the management CR:

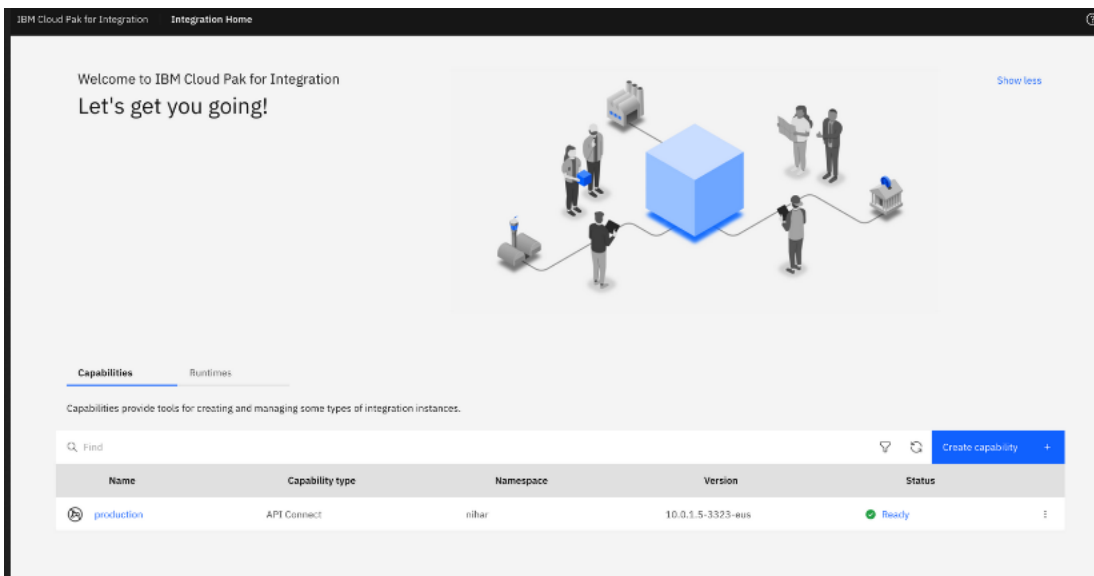
```
Status:
...
Conditions:
...
Message:          WAL Archiving is not working. Last successful archive time: 2022-07-26
20:37:14.456465 +0000 UTC. Last archive failed time: 2022-07-26 21:10:34.172539 +0000 UTC
Reason:          WALArchiveNotWorking
...
```

Note: If you see the `WAL Archiving is not working` warning message, then verify your S3 backup settings (bucket name, folder name, credentials, certificates), gather logs and open a support request.

- **VERSION:** API Connect version specified in the CR's `spec.version` setting.
- **RECONCILED VERSION:** Current API Connect version.
- **AGE:** Age of the `apiconnectcluster` CR.

Monitoring the API Connect cluster using the Platform UI

1. Start the Platform UI.
2. Click the Integration instances tab.
3. Locate the "API Connect" capability that uses the instance name that you specified as the Name during installation.
A healthy API Connect deployment displays "Ready" in the Status column



4. Click the status value to display information about the current conditions, as in the following image:

Type	Status	Updated	Message
Warning	False	a day ago	
Ready	True	15 hours ago	7/7
Pending	False	15 hours ago	
Error	False	a day ago	

A healthy deployment displays "True" for the "Ready" status.

During a fresh install, an upgrade or a day-2 operation (such as a Management backup S3 configuration change), the `apiconnectcluster` CR status might change to a different state depending on the action, but ultimately it should always reach the **Ready** status.

Monitoring Postgres disk usage on Cloud Pak for Integration

In API Connect, you can monitor the disk space that is used by the Postgres database in the Management subsystem.

The Postgres database is the core database used in the Management subsystem. It is important to monitor the Postgres disk usage to avoid running out of space and causing an outage in your deployment.

In API Connect 10.0.3 and later, the `APIConnect` operator tracks the current disk usage of the Postgres components, and regularly updates the `ManagementCluster` CR's status. When one or more of the Postgres components occupy 50% of the PVC (persistent volume claim) capacity, the `APIConnect` operator changes the CR's status from **Running** to **Warning**.

Important: If the Postgres disk usage reaches 80%, then the `APIConnect` operator brings down Postgres. If this situation occurs, contact IBM Support.

Note that there are various types of storage classes which can be used to deploy Management subsystem; for example, `local-storage` or `ceph block`. When `local-storage` is used, the entire disk is allocated to the worker node. In some cases, before the `APIConnect` operator reacts to an 80% usage condition, Kubernetes itself might face disk pressure and start evicting pods. In this situation, you might want to increase the size of the disk allocated to the worker node as explained in [Recovering on OpenShift and Cloud Pak for Integration when disks are filled by the management database](#).

Viewing the current disk usage

The `ManagementCluster` instance includes the `.status.postgresDataStats` field, where the operator displays the current disk usage of Postgres components. Run the following command to get the disk usage:

```
ocp get mgmt ml -n APIC_namespace -o json | jq .status.postgresDataStats
```

The response looks like the following example:

```
[
  {
    "instanceName": "m1-ed60c42d-postgres",
    "podName": "m1-ed60c42d-postgres-86766f69cb-xs7t5",
    "pvcCapacity": 250181844992,
    "pvcName": "m1-ed60c42d-postgres",
    "pvcType": "PostgreSQL",
    "pvcUsed": 60895232,
    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "m1-ed60c42d-postgres-backrest-shared-repo",
    "podName": "m1-ed60c42d-postgres-backrest-shared-repo-859484b5f6-r7cv6",
    "pvcCapacity": 250181844992,
    "pvcName": "m1-ed60c42d-postgres-pgbr-repo",
    "pvcType": "pgBackRest",
    "pvcUsed": 16203776,
    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "m1-ed60c42d-postgres",
    "podName": "m1-ed60c42d-postgres-86766f69cb-xs7t5",
    "pvcCapacity": 250181844992,
    "pvcName": "m1-ed60c42d-postgres-wal",
    "pvcType": "WAL",
    "pvcUsed": 201338880,
    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "m1-ed60c42d-postgres-fqbg",
    "podName": "m1-ed60c42d-postgres-fqbg-84869d976c-49qgc",
    "pvcCapacity": 250181844992,
    "pvcName": "m1-ed60c42d-postgres-fqbg",
    "pvcType": "PostgreSQL",
    "pvcUsed": 59994112,
    "pvcUsedPercentage": 0
  },
  {
    "instanceName": "m1-ed60c42d-postgres-fqbg",
    "podName": "m1-ed60c42d-postgres-fqbg-84869d976c-49qgc",

```

```

"pvcCapacity": 250181844992,
"pvcName": "m1-ed60c42d-postgres-fqbg-wal",
"pvcType": "WAL",
"pvcUsed": 201334784,
"pvcUsedPercentage": 0
},
{
"instanceName": "m1-ed60c42d-postgres-cimo",
"podName": "m1-ed60c42d-postgres-cimo-5769868b75-5z7ln",
"pvcCapacity": 250181844992,
"pvcName": "m1-ed60c42d-postgres-cimo",
"pvcType": "PostgreSQL",
"pvcUsed": 59994112,
"pvcUsedPercentage": 0
},
{
"instanceName": "m1-ed60c42d-postgres-cimo",
"podName": "m1-ed60c42d-postgres-cimo-5769868b75-5z7ln",
"pvcCapacity": 250181844992,
"pvcName": "m1-ed60c42d-postgres-cimo-wal",
"pvcType": "WAL",
"pvcUsed": 201334784,
"pvcUsedPercentage": 0
}
}
]

```

Warning condition is populated at 50% usage

When one of the Postgres components uses 50% of its allocated space, the operator changes the overall status of the **ManagementCluster** CR and the **APIConnectCluster** CR to **Warning** with an appropriate warning message. The operator also updates the **postgresDataStats** section with the current data usage of the Postgres components.

Attention: If you encounter the **Warning** condition, contact IBM Support for help correcting the root cause. To view the status from the command line, run the following command:

- **APIConnectCluster** CR:

```
oc get apiconnect
```

The following example response shows the warning status:

```

NAME          READY          RECONCILED VERSION  AGE
production    Current WAL disk usage of production-mgmt-31be1757-postgres is 60 percent. DATABASE SHUTDOWN starts at 80
percent utilization. Please contact IBM Support immediately.  Warning  10.0.1.5-eus  10.0.1.5-3394-eus  11d

```

- **ManagementCluster** CR:

```
oc get mgmt
```

The following example response shows the warning status:

```

NAME          READY  STATUS  VERSION  RECONCILED VERSION  AGE
production-mgmt  18/18  Warning  10.0.1.5-eus  10.0.1.5-3394-eus  43h

```

```
Current WAL disk usage of m1-ed60c42d-postgres is 51 percent. DATABASE SHUTDOWN starts at 80 percent utilization. Please contact IBM Support immediately
```

To view the status in IBM Cloud Pak Platform UI, click the Integration instances tab to see the status displayed next to the instance name

The screenshot shows the IBM Cloud Pak Platform UI. At the top, it says "Welcome to IBM Cloud Pak for Integration" and "Let's get you going!". Below this is a navigation bar with "Capabilities" and "Runtimes" tabs. A search bar and a "Create capability" button are also visible. The main content area displays a table of integration instances:

Name	Capability type	Namespace	Version	Status
production	API Connect	nihar	10.0.1.5-3394-eus	Warning

Click the status value (Warning, for this example) to display the Conditions list where a detailed message explains why the condition was set:

Type	Status	Updated	Message
Warning	True	6 minutes ago	Current WAL disk usage of production-mgmt-31be1757-postgres is 60 percent. DATABASE SHUTDOWN starts at 80 percent utilization. Please contact IBM Support immediately
Ready	False	6 minutes ago	
Pending	False	an hour ago	
Error	False	6 minutes ago	

Error condition is populated at 80% usage

When one of the Postgres components uses 80% of its allocated space, the operator changes the overall status of the **ManagementCluster** to **Error** and brings down Postgres avoid problems that can occur if the disk becomes completely filled.

Attention: If you encounter the **Error** condition, contact IBM Support for help correcting the root cause. To view the status from the command line, run the following command:

- **APIConnectCluster** CR:

```
oc get apiconnect
```

The following example response shows the error status:

```

NAME          READY
STATUS  VERSION      RECONCILED VERSION  AGE
production  Current disk usage is more than 80 percent. DATABASE will be shutdown. Please contact IBM Support immediately.  Error  10.0.1.5-eus  10.0.1.5-3394-eus  11d

```

- **ManagementCluster** CR:

```
oc get mgmt
```

The following example response shows the warning status:

```

NAME          READY  STATUS  VERSION      RECONCILED VERSION  AGE
production-mgmt  9/9    Error   10.0.1.5-eus  10.0.1.5-3394-eus  11d

```

To view the status in the Platform UI, click the Integration instances tab to see the status displayed next to the instance name

Welcome to IBM Cloud Pak for Integration
Let's get you going!

Capabilities | Runtimes

Capabilities provide tools for creating and managing some types of integration instances.

Find [Create capability](#)

Name	Capability type	Namespace	Version	Status
production	API Connect	nihar	10.0.1.5-3394-eus	Error

Click the status value (Error in this example) to display the Conditions list where a detailed message explains why the condition was set:

Type	Status	Updated	Message
⚠ Warning	False	10 days ago	
✔ Ready	False	11 days ago	
○ Pending	False	2 days ago	
❌ Error	True	2 days ago	Current disk usage is more than 80 percent. DATABASE will be shutdown. Please contact IBM Support immediately.

Obtaining simple health check data of Developer Portal sites by using a REST API call

Call a simple health check API from your external load balancer to dynamically determine whether a specific Developer Portal site in a cluster is working. This API call can be used by a load balancer to help determine where to route traffic.

About this task

You can use the site health REST API to determine whether a particular Developer Portal site is running. The site health API returns the current system time of the site if both the database and web server are running. This API is fast and puts no load on the system, so it is ideal for use with load balancers to help them determine where to route traffic.

Procedure

To call the site health REST API, append `/health` to the end of your Developer Portal site URL in your web browser, as follows:

```
site_url/health
```

Where `site_url` is the URL of the Developer Portal site that you want to check.

If both the database and web server of the site are running, the web browser returns the current system time. For example:

```
1511367695
```

If either the database or the web server of the site is not running, the web browser returns an error that the site can't be reached.

Logging

Configure logging for your deployment on Kubernetes.

See:

- [Gathering post-mortem logs](#)
The `generate_postmortem.sh` script gathers all logs for troubleshooting and diagnostics.
- [Changing logging levels](#)
You can enable logging for entry and exit trace and for large payloads for apim-v2 pods.

Gathering post-mortem logs

The `generate_postmortem.sh` script gathers all logs for troubleshooting and diagnostics.

About this task

When you contact IBM Support, API Connect pod logs and associated Kubernetes or OpenShift environment data are normally required to help in diagnostics. The `generate_postmortem.sh` script gathers all the required logs and data.

The logs that are gathered by the `generate_postmortem.sh` script include the logs of the current running API Connect containers. The container logs have a maximum size, and older log messages are not retained when the maximum size is reached. To ensure that the logs gathered cover the time when the problem occurred, it is recommended to reproduce the problem and then immediately gather the logs. Also, to ensure that you have logs of previous failed containers, the recommended best practice is to offboard your logs to a remote server for long-term storage. Use ELK, or another logging infrastructure, to gather logs that include terminated pods and processes. For more information, see <https://www.elastic.co/elk-stack>

Procedure

1. Download the v10 `generate_postmortem.sh` script from <https://github.com/ibm-apiconnect/v10-postmortem>.
2. Run the script in your Kubernetes or OpenShift environment, following the instructions on <https://github.com/ibm-apiconnect/v10-postmortem>.
3. Upload the generated output file with your IBM support case.

What to do next

If the logs do not cover far enough back in time, or conversely if they are too large, adjust the log retention size. For further information, see: <https://docs.docker.com/config/containers/logging/json-file/>.

If you are not able to run the `generate_postmortem.sh` script, the API Connect container logs can be gathered manually with the following command:

```
kubectl logs -n <namespace> <pod name> [-c <container name>]
```

Redirect the output of the `logs` command to a file that you can upload to the support case, for example:

```
kubectl logs -n apic-ns apic-apim-77fdf47c55-vgnr2 > apimpod.log
```

Changing logging levels

You can enable logging for entry and exit trace and for large payloads for apim-v2 pods.

About this task

For apim-v2 pods, the default logging settings do not include entry and exit trace and logging of large payloads.

Logging of large payloads is typically needed if you are logging `apim:webhookPayload` or other processes that log large variables.

You can also set the debug level without needing to restart the pods.

You can enable the settings through either the APIM deployment YAML file, the toolkit, or the REST APIs. Note, you must use the toolkit or REST API to set the debug level without restarting the pods.

Procedure

- Updating the settings in the APIM deployment YAML file

Note: This action causes a pod restart.

- The `DEBUG` variable works the same as prior releases.

```
- env:
  - name: DEBUG
    value:
audit,bhendi:error,bhendi:probe,bhendi:flags,apim:server,apim:error,apim:routes:*,apim:routesc:*,apim:oidc,apim:oidc:*
```

To enable entry and exit trace, add `trace:*` to the start of the debug string:

```
- env:
  - name: DEBUG
    value: trace:*,audit,bhendi:error,bhendi:probe,bhendi:flags,bhendi:webhookAudit,bhendi:cassandra-
transactions,apim:server,apim:error,apim:routes:*,apim:routesc:*,apim:oidc,apim:oidc:*,apim:taskmanager:info
```

- To enable large object logging:

```
- env:
  - name: VELOX_LOG_FULL_PAYLOAD
    value: "true"
```

- Updating the settings by using the toolkit

Note: This action does not cause a pod restart. Also, this action is for a single APIM pod environment only.

```
apic log-spec:get
apic log-spec:update LOG_SPEC_FILE
```

Where the spec file contains:

```
{"specification": "apim:routes:log_spec,audit,bhendi:error,apim:server,apim:error", "large_objects": true}
```

- Updating the settings by using the REST API

Note: This action does not cause a pod restart. Also, this action is for a single APIM pod environment only.

```
GET /cloud/api/log-spec
PUT /cloud/api/log-spec
```

For example, to set debug level and large objects logging:

```
curl -k https://172.16.140.212:3003/api/cloud/log-spec -X PUT -H "Authorization: Bearer $BEARER"
-H "Accept: application/json" -d '{"specification":
"apim:routes:log_spec,audit,bhendi:error,apim:server,apim:error", "large_objects": true}'
-H "Content-Type: application/json"
```

The command responds with:

```
{
  "specification": "apim:routes:log_spec,audit,bhendi:error,apim:server,apim:error",
  "large_objects": true,
  "message": "Successfully changed the log specification on all apim-v2 pods. Success on IP(s): 172.16.140.212,
172.16.140.213"
}
```


Note: The response might include the following warning, which can be ignored:

```
WARNING: Could not change the log specification on all apim-v2 pods. Success on IP(s): 4.5.6.7 Failed on IP(s): 1.2.3.4
```

To view the REST APIs for logging, go to [API Connect REST APIs](#), and select IBM API Connect Platform - Cloud Management API 2.0.0 reference_>_Resource: Log Spec.

Troubleshooting

Troubleshoot API Connect subsystems on Kubernetes.

Note: In the  Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
See:

- [Troubleshooting on Kubernetes](#)
Review the following topics for help in resolving issues with your API Connect deployment on Kubernetes.
- [Troubleshooting on OpenShift and Cloud Pak for Integration](#)
Review the following topics for help in resolving issues with your API Connect deployment on OpenShift or Cloud Pak for Integration.

Troubleshooting on Kubernetes

Review the following topics for help in resolving issues with your API Connect deployment on Kubernetes.

- [Troubleshooting the management subsystem](#)
Troubleshoot problems with the management subsystem and management subsystem database
- [Troubleshooting Analytics on Kubernetes](#)
If your API Connect Analytics subsystem encounters problems, review the following topics for help resolving the issue.

Troubleshooting the management subsystem

Troubleshoot problems with the management subsystem and management subsystem database

See:

- [Troubleshooting the management database](#)
You can troubleshoot the API Connect management database
- [Increasing the PostgreSQL archive timeout on Kubernetes](#)
Increase the `archive_timeout` to 3600 (seconds) so that accumulated wal files do not cause a disk-space issue during a network outage.
- [Recovering when disks are filled by the management database](#)
Resize a PersistentVolumeClaim (PVC) to recover when disks are filled by the management database.

Troubleshooting the management database

You can troubleshoot the API Connect management database

Database Replica Pods stuck in Unknown or Pending state

In certain scenarios, a postgres replica pod may not recover to a healthy state when a restore completes, a node outage occurs, or after a fresh install or upgrade. In these cases, a postgres pod remains in a **Unknown** or a **Pending** state after a number of minutes. The pod fail to get into a **Running** state.

This situation occurs when the replicas do not initialize properly. You can use the `patronictl reinit` command to reinitialize the replica. Note that this command syncs the replica's volume data from the current Primary pod.

Use the following steps to get the pod back into a working state:

1. Exec onto the failing pod:

```
kubectl exec -it <postgres_replica_pod_name> -n <namespace> -- bash
```

2. List the cluster members:

```
patronictl list
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) -----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| fxpk-management-01191b80-postgres-586f899fdf-6s25b | 172.16.172.244 | | start failed | | unknown |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51 | Leader | running | 3 | |
| fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m | 172.16.53.68 | | running | 3 | 0 |
+-----+-----+-----+-----+-----+-----+-----+
```

In the example shown above `fxpk-management-01191b80-postgres-586f899fdf-6s25b` is not in running state.

Note the `clusterName` and `replicaName` which are not up:

- `clusterName` - `fxpk-management-01191b80-postgres`
- `replicaName` - `fxpk-management-01191b80-postgres-586f899fdf-6s25b`

3. Run:

```
patronictl reinit <clusterName> <replicaName-which-is-not-running>
```

Example:

```
patronictl reinit fxpk-management-01191b80-postgres fxpk-management-01191b80-postgres-586f899fdf-6s25b
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) -----+-----+-----+-----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| fxpk-management-01191b80-postgres-586f899fdf-6s25b | 172.16.172.244 | | start failed | | unknown |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51 | Leader | running | 3 | |
| fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m | 172.16.53.68 | | running | 3 | 0 |
+-----+-----+-----+-----+-----+-----+
Are you sure you want to reinitialize members fxpk-management-01191b80-postgres-586f899fdf-6s25b? [y/N]: y
Success: reinitialize for member fxpk-management-01191b80-postgres-586f899fdf-6s25b
```

4. Run `patronictl list` again.

You may also observe that the replica is on a different Timeline (TL) and possibly have a Lag in MB. It may take a few minutes for the pod to switch onto the same TL as the others and the Lag should slowly go to 0.

For example:

```
bash-4.2$ patronictl list
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) -----+-----+-----+-----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| fxpk-management-01191b80-postgres-586f899fdf-6s25b | 172.16.172.244 | | running | 1 | 23360 |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51 | Leader | running | 3 | |
| fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m | 172.16.53.68 | | running | 3 | 0 |
+-----+-----+-----+-----+-----+-----+
```

5. The pod that previously was in an `Unknown` or `Pending` state or `(0/1) Running` state is now in `(1/1) Running` state.

Increasing the PostgreSQL `archive_timeout` on Kubernetes

Increase the `archive_timeout` to 3600 (seconds) so that accumulated wal files do not cause a disk-space issue during a network outage.

About this task

PostgreSQL sends its wal archives to the backup location (S3 or local PVC) whenever the configured `archive_timeout` period elapses. By default, the `archive_timeout` is set to 60 seconds. During a network outage, the files cannot be transferred and instead accumulate on the current server, and could cause a disk-space issue. Increasing the `archive_timeout` value helps to mitigate this issue by creating archive files less frequently.

For information on tracking the Postgres disk space, see [Monitoring Postgres disk usage](#).

Complete the following steps to increase the PostgreSQL `archive_timeout` value.

Procedure

1. Get the name of the `pgha` config map by running the following command:

```
kubectl -n <namespace> get cm -l pgha-config='true'
```

The response looks like the following example:

```
NAME                               DATA  AGE
apis-produ-d18f3f0-0d18f3f0-postgres-pgha-config  5      22d
```

2. Edit the config map by running the following command:

```
kubectl -n <namespace> edit configmap <config-map-name>
```

In the command, replace `<config-map-name>` with the `NAME` returned in step 1.

3. Change the `archive_timeout` value to 3600.

The value must be an integer, and represents the number of seconds that must elapse before the archive file is generated.

4. Save the file and exit the editor.

The configuration is updated automatically; archives will be generated and pushed every 3600 seconds.

Recovering when disks are filled by the management database

Resize a PersistentVolumeClaim (PVC) to recover when disks are filled by the management database.

There are two methods to resize a PVC:

- Expand the PVC.
- Create a new, larger PVC, and copy the PVC contents.

Expansion of the PVC is easier and quicker than creating a new PVC and copying contents. However, some storage classes do not support PVC volume expansion. For these classes, you must create a new PVC.

Note: If you are using local storage, the only way to support increased data size is to add a disk

1. Determine if the StorageClass which provisioned the PVC supports VolumeExpansion. You can only expand a PVC if its storage class `allowVolumeExpansion` field is set to `true`.

```
$ kubectl get sc
NAME                PROVISIONER
rook-ceph-block     rook-ceph.rbd.csi.ceph.com

$ kubectl describe sc/rook-ceph-block | grep "AllowVolumeExpansion"
AllowVolumeExpansion: True
```

2. Choose a resizing method for the storage class:
 - If `AllowVolumeExpansion` is `true`, go to [Resizing a PersistentVolumeClaim for Postgres by volume expansion](#)
 - If `AllowVolumeExpansion` is not `true`, you cannot expand the PVC. Go to [Resizing a PersistentVolumeClaim for Postgres by moving content to a new PVC](#).
- [Resizing a PersistentVolumeClaim for Postgres by volume expansion](#)
Resize a PersistentVolumeClaim (PVC) for Postgres that supports volume expansion, on native Kubernetes.
- [Resizing a PersistentVolumeClaim for Postgres by moving content to a new PVC](#)
Resize a PersistentVolumeClaim (PVC) for Postgres by creating a new, larger PVC, and moving content between the PVCs, on native Kubernetes.

Resizing a PersistentVolumeClaim for Postgres by volume expansion

Resize a PersistentVolumeClaim (PVC) for Postgres that supports volume expansion, on native Kubernetes.

Before you begin

Determine whether your storage class supports volume expansion. See [Recovering when disks are filled by the management database](#).

About this task

You can expand a PVC if your storage provisioner supports volume expansion. This procedure is for use on native Kubernetes.

Note: This procedure is for use only with Postgres.

Procedure

1. Determine the primary Postgres pod.

```
kubectl get pods --selector role=master | grep -v bootstrapselector=role=master
```

2. Gracefully shutdown Postgres pods.

- a. Exec into the Postgres primary pod.

```
kubectl exec -it <postgres-pod> -- bash
```

- b. Pause Patroni process. Note: you can skip this step if using the one replica profile.

```
patronictl pause
```

- c. Stop the Postgres process gracefully.

```
pg_ctl stop -D /pgdata/<cluster-name>
```

For example:

- `cluster-name` is usually the name of the cluster. If primary pod name is `m1-1b12bf81-postgres-58dfcc5f97-fsvvt`, `cluster-name` is `m1-1b12bf81-postgres`.
- There will be a folder named `cluster-name` located under `/pgdata/`.
- In this example, the command is:

```
pg_ctl stop -D /pgdata/m1-1b12bf81-postgres
```

3. Determine the PVC linked to the primary Postgres pod, and obtain the `claimName` of the PVC.

```
kubectl get pod <postgres-pod> -o json | jq '.spec.volumes[] | select ( has ("persistentVolumeClaim"))'
```

4. Scale down the Postgres deployment.

```
kubectl get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
```

If you are using `n3` profile then you find 3 deployments. Scale them down:

```
kubectl scale deploy <deployment-name> --replicas=0
```

5. Edit the PVC you want to increase.

```
kubectl edit pvc <claimName>
```

6. Update `spec.resources.requests.storage` field value with new size. For example:

```
spec:
  resources:
    requests:
      storage: 20Gi
```

7. Scale up the Postgres deployment.

```
kubectl get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
```

If you are using the `n3` profile then you find 3 deployments. Scale them up:

```
kubectl scale deploy <deployment-name> --replicas=1
```

8. If using the `n3` profile, resume the Patroni process. If using the `n1` profile, skip this step.

a. Determine the new primary Postgres pod.

```
kubectl get pods --selector role=master | grep -v bootstrapselector=role=master
```

b. Exec into the new Postgres primary pod.

```
kubectl exec -it <postgres-pod> -- bash
```

c. Resume Patroni.

```
patronictl resume
```

Resizing a PersistentVolumeClaim for Postgres by moving content to a new PVC

Resize a PersistentVolumeClaim (PVC) for Postgres by creating a new, larger PVC, and moving content between the PVCs, on native Kubernetes.

Before you begin

Determine whether your storage class supports volume expansion. See [Recovering when disks are filled by the management database](#).

About this task

Use this procedure when the storage class does not support volume expansion. This procedure is for use on native Kubernetes.

Note: This procedure is for use only with Postgres.

Procedure

1. Create a new temporary PVC. Ensure the requested storage size is more than the original PVC. For example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: temp-pvc
spec:
  storageClassName: rook-ceph-block
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

2. Determine the primary Postgres pod.

```
kubectl get pods --selector role=master | grep -v bootstrapselector=role=master
```

3. Gracefully shutdown Postgres pods.

a. Exec into the Postgres primary pod.

```
kubectl exec -it <postgres-pod> -- bash
```

b. Pause the Patroni process. Note: you can skip this step if using the one replica profile.

```
patronictl pause
```

c. Stop the Postgres process gracefully.

```
pg_ctl stop -D /pgdata/<cluster-name>
```

For example:

- `cluster-name` is usually the name of the cluster. If primary pod name is `m1-1b12bf81-postgres-58dfcc5f97-fsvvt`, `cluster-name` is `m1-1b12bf81-postgres`.
- There will be a folder named `cluster-name` located under `/pgdata/`.
- In this example, the command is:

```
pg_ctl stop -D /pgdata/m1-1b12bf81-postgres
```

- Determine the PVC linked to the primary Postgres pod, and obtain the `claimName` of the PVC.

```
kubectl get pod <postgres-pod> -o json | jq '.spec.volumes[] | select ( has ("persistentVolumeClaim"))'
```

- Scale down the Postgres deployment.

```
kubectl get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
```

If you are using `n3` profile then you find 3 deployments. Scale them down:

```
kubectl scale deploy <deployment-name> --replicas=0
```

- Start a BusyBox pod with the source and destination PVC volumes mounted, to swap the content between PVCs. For example:

```
apiVersion: v1
kind: Pod
metadata:
  name: swap-pvc
spec:
  volumes:
    - name: volume-source
      persistentVolumeClaim:
        claimName: m1-mgmt-35817114-postgres-wal
    - name: volume-destination
      persistentVolumeClaim:
        claimName: temp-pvc
  containers:
    - name: debug
      image: busybox
      command: ['sleep', '3600']
      volumeMounts:
        - name: volume-source
          mountPath: /data-source
        - name: volume-destination
          mountPath: /data-destination
```

- Exec to the BusyBox pod, and

```
kubectl exec -it <busybox-pod-name> -- sh
```

```
cp -r data-source/ data-destination/
```

- Copy content to the temporary PVC. For example, using the `mountPath` values shown in Step 6:

```
cp -r data-source/ data-destination/
```

- Delete the BusyBox pod.

```
kubectl delete pod <busybox-pod-name>
```

- Delete the original PVC.

```
kubectl delete pvc <claimName>
```

- Create an original PVC with the same name but with increased size. For example:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: m1-mgmt-35817114-postgres-wal
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: rook-ceph-block
  volumeMode: Filesystem
```

- Start the BusyBox pod again and this time copy the content from destination to source PVC.

```
kubectl exec -it <busybox-pod> -- sh
```

```
cp -r data-destination/ data-source/
```

- Delete the BusyBox pod.

```
kubectl delete pod <busybox-pod>
```

- Delete the temporary PVC that you created for swapping the content.

```
kubectl delete pvc temp-pvc
```

- Scale up the Postgres deployment.

```
kubectl get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
```

If you are using the `n3` profile then you find 3 deployments. Scale them up:

```
kubectl scale deploy <deployment-name> --replicas=1
```

- If using the `n3` profile, resume the Patroni process. If using the `n1` profile, skip this step.

- Determine the new primary Postgres pod.


```
kubectl get pods --selector role=master | grep -v bootstrapselector=role=master
```

b. Exec into the new Postgres primary pod.

```
kubectl exec -it <postgres-pod> -- bash
```

c. Resume Patroni.

```
patronictl resume
```

Troubleshooting Analytics on Kubernetes

If your API Connect Analytics subsystem encounters problems, review the following topics for help resolving the issue.

- [Analytics offload issues on Kubernetes](#)
If your Analytics offload fails, you can perform checks to help determine the cause. Common causes for an offload failure are connection or security-related issues.
- [Empty Analytics display on Kubernetes](#)
If your API Connect analytics data doesn't display in the Analytics dashboard, complete the following steps to determine the cause of the problem.
- [Analytics running out of disk space on Kubernetes](#)
If your Analytics data is taking up too much disk space, complete the following steps to resolve the problem.

Analytics offload issues on Kubernetes

If your Analytics offload fails, you can perform checks to help determine the cause. Common causes for an offload failure are connection or security-related issues.

When you save the configuration files, their syntax is first validated to verify that it can be applied. However, this does not confirm that other parts of your analytics collection ecosystem are trouble-free, for example, that your endpoints are running, or that the TLS certificates are validated. Errors in these areas may also result in no data output.

To verify issues you will need to examine the logs of your analytics ingestion pods. (Note that it may take a few minutes for the pods to pick up changes in the configuration files after edits are applied.)

For information on obtaining logs, refer to [Logging](#).

Following are some common checks you can make to troubleshoot the offload:

- Verify that the specified URL or the host and port values for the target system are accurate.
- Verify that the server or cluster of servers in the target system are running and reachable. If applicable, also verify that the connection problems are not caused by firewall settings.
- If you are using Transport Layer Security (TLS) to establish a private and secure communication channel to the target system, investigate whether the issue is caused by SSL certificate errors; for example, untrusted server certificates, certificates that have expired or are not yet valid, or missing intermediate (or chain) certificates. For more information on using TLS, see [Creating a TLS profile](#).
- If offloading to an Elasticsearch or Kafka target, verify that your configuration settings are valid; for example, the user credentials for authenticating to Elasticsearch or the Kafka topic name.

Empty Analytics display on Kubernetes

If your API Connect analytics data doesn't display in the Analytics dashboard, complete the following steps to determine the cause of the problem.

Procedure

1. Run the following command to check the logs of the mtls-gw pod:

```
kubectl logs analyt-mtls-gw-7665f5d445-4cx4n -n <APIC_namespace>
```

The response looks like the following example:

```
10.254.6.1 - - [22/Sep/2021:18:05:54 +0000] "POST /x2020/v1/apievent/_bulk HTTP/1.1" 200 2 "-" "-" 2841 0.072 0.070
```

- a. Check the response for any `_bulk` request coming from the gateway, as shown in the example.
If there are no `_bulk` requests, then the problem is with the gateway and the wsm-agent, which is not sending bulk requests to the Analytics server.
- b. If you see the `_bulk` request., then check the return code in the response.
If `_bulk` requests are processed successfully you will see a return code 200 (as in the example), which indicates that the bulk request has been ingested.
- c. If it is any other return code, check the ingestion pod logs for any errors, which will indicate any issues during ingestion:

```
kubectl logs analyt-ingestion-78d7cff7f7-6pktd
```

2. If everything with the data flow looks fine, and if you are unable to see analytics data, then check the health of Analytics using the CLI.

- a. If you do not have the CLI installed see [./com.ibm.apic.apionprem.doc/analytics_use_cli_and_rest_api.html](#) for the steps on installing and using.
- b. Run the following CLI command:

```
apic --mode analytics clustermgmt:getHealth --server <management server platform api endpoint> --analytics-service <analytics service name> --format json
```

where:

- `<management server platform api endpoint>` is the Management server admin endpoint.
- `<analytics service name>` is the name of Analytics service as configured in the Cloud Manager.

When you check the cluster's health, the response for a single node analytics deployment looks like the following example:

```
{
  "cluster_name": "apic-analytics-cluster",
  "status": "yellow",
  "timed_out": false,
  "number_of_nodes": 1,
  "number_of_data_nodes": 1,
  "discovered_master": true,
  "discovered_cluster_manager": true,
  "active_primary_shards": 8,
  "active_shards": 8,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 12,
  "delayed_unassigned_shards": 0,
  "number_of_pending_tasks": 0,
  "number_of_in_flight_fetch": 0,
  "task_max_waiting_in_queue_millis": 0,
  "active_shards_percent_as_number": 40
}
```

Note: Single node analytics clusters never show **green** in the health output. Green indicates that the data is replicated to at least one other node, and since there are no other nodes in a single node cluster it remains as **yellow**.

- Check the status of the cluster (red, yellow, green) in the response.
In the example, the status is set to yellow.

- Green indicates all indices are fine.
- Yellow indicates there are some unassigned shards, which might still be replicating.
- Red indicates there is a problem with one or more of the indices.

- Run the following command to get a list of indices:

```
apic --mode analytics clustermgmt:catIndices --server [SERVER] --analytics-service [ANALYTICS SERVICE]
```

where:

- `SERVER` is the Management server admin endpoint.
- `ANALYTICS SERVICE` is the name of Analytics service as configured in the Cloud Manager.

- From the list of indices returned, identify the those with status of 'red'.

- Delete each red index by running the following command:

```
apic --mode analytics clustermgmt:deleteIndex --index <index_name> --server [SERVER] --analytics-service [ANALYTICS SERVICE] --format json
```

where:

- `SERVER` is the Management server admin endpoint.
- `ANALYTICS SERVICE` is the name of Analytics service as configured in the Cloud Manager.

Analytics running out of disk space on Kubernetes

If your Analytics data is taking up too much disk space, complete the following steps to resolve the problem.

Before you begin

This operation requires the use of the Analytics CLI. [Using the Analytics CLI](#).

About this task

Analytics indices taking up too much space

Procedure

- Run the following command to display the indices:

```
apic --mode analytics clustermgmt:catIndices --server [SERVER] --analytics-service [ANALYTICS SERVICE]
```

where:

- `SERVER` is the Management server admin endpoint.
- `ANALYTICS SERVICE` is the name of Analytics service as configured in the Cloud Manager.

- Find the old and large indices that can be deleted, and delete each index using the following command:

```
apic --mode analytics clustermgmt:deleteIndex --index <index_name> --server [SERVER] --analytics-service [ANALYTICS SERVICE] --format json
```

where:

- `SERVER` is the Management server admin endpoint.
- `ANALYTICS SERVICE` is the name of Analytics service as configured in the Cloud Manager.

- Reduce the retention period: [Configuring data retention and index rollover time periods](#)

- Apply a filter to reduce the index size by excluding large fields, such as `response_body`, from being saved in the index.

For example:

```

filter {
  if "apicapievent" in [tags] {
    mutate {
      remove_field => ["response_body"]
    }
  }
}

```

For information on creating filters for Analytics data, see [Modifying incoming analytics data](#).

Troubleshooting on OpenShift and Cloud Pak for Integration

Review the following topics for help in resolving issues with your API Connect deployment on OpenShift or Cloud Pak for Integration.

- [Power-cycling check on OpenShift and Cloud Pak for Integration](#)
If you cycle the power on an API Connect cluster that is deployed on OpenShift or Cloud Pak for Integration, you might need to reset the IP address for the `kubernetes.io/hostname`.
- [Troubleshooting the Management subsystem on OpenShift and Cloud Pak for Integration](#)
Troubleshoot problems with the Management subsystem and the management subsystem database.
- [Troubleshooting Analytics on OpenShift and Cloud Pak for Integration](#)
If your API Connect Analytics subsystem encounters problems, review the following topics for help resolving the issue.

Power-cycling check on OpenShift and Cloud Pak for Integration

If you cycle the power on an API Connect cluster that is deployed on OpenShift or Cloud Pak for Integration, you might need to reset the IP address for the `kubernetes.io/hostname`.

About this task

You will know there is a problem if you cycle the power for the cluster and receive an error, or find that the deployment remains in the **pending** state indefinitely because some of the pods failed to start. Complete the following steps to resolve the problem:

Procedure

1. Run the following command to check the label on the worker nodes:

```
oc get nodes -l kubernetes.io/hostname=localhost
```

If the response shows the following message, then the power cycle did not cause a problem and you can skip the rest of this task:

```
No resources found
```

If the response looks like the following example (the IP address will be different), proceed to the next step:

NAME	STATUS	ROLES	AGE	VERSION
ip-233.252.0.0.us-west-1.compute.internal	Ready	worker	27h	v1.19.0+8d12420

2. Run the following command to correct the problem:

```
for i in $(oc get nodes -l kubernetes.io/hostname=localhost|grep worker|awk '{print $1}'); do echo $i; oc label nodes $i kubernetes.io/hostname=`echo $i|awk -F'.' '{print $1}'` --overwrite=true; done
```

Troubleshooting the Management subsystem on OpenShift and Cloud Pak for Integration

Troubleshoot problems with the Management subsystem and the management subsystem database.

See:

- [Troubleshooting the management database on OpenShift and Cloud Pak for Integration](#)
You can troubleshoot the API Connect management database by complete the following tasks.
- [Increasing the PostgreSQL archive timeout on OpenShift and Cloud Pak for Integration](#)
Increase change the `archive_timeout` to 3600 (seconds) so that accumulated wal files do not cause a disk-space issue during a network outage.
- [Recovering on OpenShift and Cloud Pak for Integration when disks are filled by the management database](#)
Resize a PersistentVolumeClaim (PVC) on OpenShift to recover when disks are filled by the API Connect management database.

Troubleshooting the management database on OpenShift and Cloud Pak for Integration

You can troubleshoot the API Connect management database by complete the following tasks.

Database Replica Pods stuck in Unknown or Pending state

In certain scenarios, a postgres replica pod may not recover to a healthy state when a restore completes, a node outage occurs, or after a fresh install or upgrade. In these cases, a postgres pod remains in a **Unknown** or a **Pending** state after a number of minutes. The pod fail to get into a **Running** state.

This situation occurs when the replicas do not initialize properly. You can use the `patronictl reinit` command to reinitialize the replica. Note that this command syncs the replica's volume data from the current Primary pod.

Use the following steps to get the pod back into a working state:

1. Exec onto the failing pod:

```
oc exec -it <postgres_replica_pod_name> -n <namespace> -- bash
```

2. List the cluster members:

```
patronictl list
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) -----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| fxpk-management-01191b80-postgres-586f899fdf-6s25b | 172.16.172.244 | | start failed | | unknown |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51 | Leader | running | 3 | |
| fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m | 172.16.53.68 | | running | 3 | 0 |
+-----+-----+-----+-----+-----+-----+
```

In the example shown above `fxpk-management-01191b80-postgres-586f899fdf-6s25b` is not in running state.

Note the `clusterName` and `replicaName` which are not up:

- `clusterName` - `fxpk-management-01191b80-postgres`
- `replicaName` - `fxpk-management-01191b80-postgres-586f899fdf-6s25b`

3. Run:

```
patronictl reinit <clusterName> <replicaName-which-is-not-running>
```

Example:

```
patronictl reinit fxpk-management-01191b80-postgres fxpk-management-01191b80-postgres-586f899fdf-6s25b
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) -----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| fxpk-management-01191b80-postgres-586f899fdf-6s25b | 172.16.172.244 | | start failed | | unknown |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51 | Leader | running | 3 | |
| fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m | 172.16.53.68 | | running | 3 | 0 |
+-----+-----+-----+-----+-----+-----+
```

```
Are you sure you want to reinitialize members fxpk-management-01191b80-postgres-586f899fdf-6s25b? [y/N]: y
Success: reinitialize for member fxpk-management-01191b80-postgres-586f899fdf-6s25b
```

4. Run `patronictl list` again.

You may also observe that the replica is on a different Timeline (TL) and possibly have a Lag in MB. It may take a few minutes for the pod to switch onto the same TL as the others and the Lag should slowly go to 0.

For example:

```
bash-4.2$ patronictl list
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) -----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| fxpk-management-01191b80-postgres-586f899fdf-6s25b | 172.16.172.244 | | running | 1 | 23360 |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51 | Leader | running | 3 | |
| fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m | 172.16.53.68 | | running | 3 | 0 |
+-----+-----+-----+-----+-----+-----+
```

5. The pod that previously was in an **Unknown** or **Pending** state or (0/1) **Running** state is now in (1/1) **Running** state.

Increasing the PostgreSQL `archive_timeout` on OpenShift and Cloud Pak for Integration

Increase change the `archive_timeout` to 3600 (seconds) so that accumulated wal files do not cause a disk-space issue during a network outage.

About this task

PostgreSQL sends its wal archives to the backup location (S3 or local PVC) whenever the configured `archive_timeout` period elapses. By default, the `archive_timeout` is set to 60 seconds. During a network outage, the files cannot be transferred and instead accumulate on the current server, and could cause a disk-space issue. Increasing the `archive_timeout` value helps to mitigate this issue by creating archive files less frequently.

For information on tracking the Postgres disk space, see [Monitoring Postgres disk usage](#).

Complete the following steps to increase the PostgreSQL `archive_timeout` value.

Procedure

1. Get the name of the `pgha` config map by running the following command:

```
oc -n <APIC_namespace> get cm -l pgha-config='true'
```

The response looks like the following example:

NAME	DATA	AGE
apis-prod-u-d18f3f0-0d18f3f0-postgres-pgha-config	5	22d

2. Edit the config map by running the following command:

```
oc -n <APIC_namespace> edit configmap <config-map-name>
```

In the command, replace `<config-map-name>` with the **NAME** returned in step 1.

3. Change the `archive_timeout` value to **3600**.

The value must be an integer, and represents the number of seconds that must elapse before the archive file is generated.

4. Save the file and exit the editor.

The configuration is updated automatically; archives will be generated and pushed every **3600** seconds.

Recovering on OpenShift and Cloud Pak for Integration when disks are filled by the management database

Resize a PersistentVolumeClaim (PVC) on OpenShift to recover when disks are filled by the API Connect management database.

There are two methods to resize a PVC:

- Expand the PVC.
- Create a new, larger PVC, and copy the PVC contents.

Expansion of the PVC is easier and faster than creating a new PVC and copying contents. However, some storage classes do not support PVC volume expansion. For these classes, you must create a new PVC.

Note: If you are using local storage, the only way to support increased data size is to add a disk.

1. Determine if the StorageClass which provisioned the PVC supports VolumeExpansion.

You can only expand a PVC if its storage class `allowVolumeExpansion` field is set to **true**.

```
oc get sc
```

NAME	PROVISIONER
rook-ceph-block	rook-ceph.rbd.csi.ceph.com

```
oc describe sc/rook-ceph-block | grep "AllowVolumeExpansion"
```

```
AllowVolumeExpansion: True
```

2. Choose a resizing method for the storage class:

- If `AllowVolumeExpansion` is **true**, see [Resizing a PVC for Postgres on OpenShift by using volume expansion](#)
- If `AllowVolumeExpansion` is not **true**, you cannot expand the PVC. See [Resizing a PVC for Postgres on OpenShift by moving content](#).
- [Resizing a PVC for Postgres on OpenShift by using volume expansion](#)
Resize a Persistent Volume Claim (PVC) for Postgres that supports volume expansion.
- [Resizing a PVC for Postgres on OpenShift by moving content](#)
Resize a Persistent Volume Claim (PVC) for Postgres on OpenShift by creating a new, larger PVC, and then moving content between the PVCs.

Resizing a PVC for Postgres on OpenShift by using volume expansion

Resize a Persistent Volume Claim (PVC) for Postgres that supports volume expansion.

Before you begin

This task only applies to the PVC used by Postgres. To determine whether your storage class supports volume expansion, see [Recovering on OpenShift and Cloud Pak for Integration when disks are filled by the management database](#).

About this task

You can expand a PVC if your storage provisioner supports volume expansion.

Procedure

1. Determine the primary Postgres pod.

```
oc get pods --selector role=master | grep -v bootstrapselector=role=master
```

2. Gracefully shutdown Postgres pods.

a. Exec into the Postgres primary pod.

```
oc exec -it <postgres-pod> -- bash
```

b. Pause Patroni process. Note: you can skip this step if using the one replica profile.

```
patronictl pause
```

c. Stop the Postgres process gracefully.

```
pg_ctl stop -D /pgdata/<cluster-name>
```

For example:

- `cluster-name` is usually the name of the cluster. If primary pod name is `m1-1b12bf81-postgres-58dfcc5f97-fsvvt`, `cluster-name` is `m1-1b12bf81-postgres`.
- There will be a folder named `cluster-name` located under `/pgdata/`.
- In this example, the command is:

```
pg_ctl stop -D /pgdata/m1-1b12bf81-postgres
```

3. Determine the PVC linked to the primary Postgres pod, and obtain the `claimName` of the PVC.

```
oc get pod <postgres-pod> -o json | jq '.spec.volumes[] | select ( has ("persistentVolumeClaim"))'
```

4. Scale down the Postgres deployment.

```
oc get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
```

If you are using `n3` profile then you find 3 deployments. Scale them down:

```
oc scale deploy <deployment-name> --replicas=0
```

5. Edit the PVC you want to increase.

```
oc edit pvc <claimName>
```

6. Update `spec.resources.requests.storage` field value with new size. For example:

```
spec:
  resources:
    requests:
      storage: 20Gi
```

7. Scale up the Postgres deployment.

```
oc get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
```

If you are using the `n3` profile then you find 3 deployments. Scale them up:

```
oc scale deploy <deployment-name> --replicas=1
```

8. If you are using the `n3` profile, resume the Patroni process. If using the `n1` profile, skip this step.

a. Determine the new primary Postgres pod.

```
oc get pods --selector role=master | grep -v bootstrapselector=role=master
```

b. Exec into the new Postgres primary pod.

```
oc exec -it <postgres-pod> -- bash
```

c. Resume Patroni.

```
patronictl resume
```

Resizing a PVC for Postgres on OpenShift by moving content

Resize a Persistent Volume Claim (PVC) for Postgres on OpenShift by creating a new, larger PVC, and then moving content between the PVCs.

Before you begin

This task only applies to the PVC used by Postgres.

About this task

Use this procedure when the storage class does not support volume expansion.

Procedure

1. Create a new temporary PVC. Ensure the requested storage size is more than the original PVC. For example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: temp-pvc
spec:
  storageClassName: rook-ceph-block
  accessModes:
    - ReadWriteOnce
  resources:
```

```
requests:
  storage: 10Gi
```

2. Determine the primary Postgres pod.

```
oc get pods --selector role=master | grep -v bootstrapselector=role=master
```

3. Gracefully shutdown Postgres pods.

a. Exec into the Postgres primary pod.

```
oc exec -it <postgres-pod> -- bash
```

b. Pause the Patroni process. Note: you can skip this step if using the one replica profile.

```
patronictl pause
```

c. Stop the Postgres process gracefully.

```
pg_ctl stop -D /pgdata/<cluster-name>
```

For example:

- `cluster-name` is usually the name of the cluster. If primary pod name is `m1-1b12bf81-postgres-58dfcc5f97-fsvvt`, `cluster-name` is `m1-1b12bf81-postgres`.
- There will be a folder named `cluster-name` located under `/pgdata/`.
- In this example, the command is:

```
pg_ctl stop -D /pgdata/m1-1b12bf81-postgres
```

4. Determine the PVC linked to the primary Postgres pod, and obtain the `claimName` of the PVC.

```
oc get pod <postgres-pod> -o json | jq '.spec.volumes[] | select ( has ("persistentVolumeClaim"))'
```

5. Scale down the Postgres deployment.

```
oc get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
```

If you are using `n3` profile then you find 3 deployments. Scale them down:

```
oc scale deploy <deployment-name> --replicas=0
```

6. Start a BusyBox pod with the source and destination PVC volumes mounted, to swap the content between PVCs. For example:

```
apiVersion: v1
kind: Pod
metadata:
  name: swap-pvc
spec:
  volumes:
    - name: volume-source
      persistentVolumeClaim:
        claimName: m1-mgmt-35817114-postgres-wal
    - name: volume-destination
      persistentVolumeClaim:
        claimName: temp-pvc
  containers:
    - name: debug
      image: busybox
      command: ['sleep', '3600']
      volumeMounts:
        - name: volume-source
          mountPath: /data-source
        - name: volume-destination
          mountPath: /data-destination
```

7. Exec to the BusyBox pod, and run the following commands:

```
oc exec -it <busybox-pod-name> -- sh
```

```
cp -r data-source/ data-destination/
```

8. Copy content to the temporary PVC. For example, using the `mountPath` values shown in Step 6:

```
cp -r data-source/ data-destination/
```

9. Delete the BusyBox pod.

```
oc delete pod <busybox-pod-name>
```

10. Delete the original PVC.

```
oc delete pvc <claimName>
```

11. Create an original PVC with the same name but with increased size. For example:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: m1-mgmt-35817114-postgres-wal
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
```

```
storageClassName: rook-ceph-block
volumeMode: Filesystem
```

12. Run the following commands to start the BusyBox pod again and copy the content from the destination to the source PVC:

```
oc exec -it <busybox-pod> -- sh

cp -r data-destination/ data-source/
```

13. Delete the BusyBox pod.

```
oc delete pod <busybox-pod>
```

14. Delete the temporary PVC that you created for swapping the content.

```
oc delete pvc temp-pvc
```

15. Scale up the Postgres deployment.

```
oc get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
```

If you are using the `n3` profile then you find 3 deployments. Scale them up:

```
oc scale deploy <deployment-name> --replicas=1
```

16. If using the `n3` profile, resume the Patroni process. If using the `n1` profile, skip this step.

a. Determine the new primary Postgres pod.

```
oc get pods --selector role=master | grep -v bootstrapselector=role=master
```

b. Exec into the new Postgres primary pod.

```
oc exec -it <postgres-pod> -- bash
```

c. Resume Patroni.

```
patronictl resume
```

Troubleshooting Analytics on OpenShift and Cloud Pak for Integration

If your API Connect Analytics subsystem encounters problems, review the following topics for help resolving the issue.

- [Analytics offload issues on OpenShift and Cloud Pak for Integration](#)
If your Analytics offload fails, you can perform checks to help determine the cause. Common causes for an offload failure are connection or security-related issues.
- [Empty Analytics display on OpenShift and Cloud Pak for Integration](#)
If your API Connect analytics data doesn't display in the Analytics dashboard, complete the following steps to determine the cause of the problem.
- [Analytics running out of disk space on OpenShift and Cloud Pak for Integration](#)
If your Analytics data is taking up too much disk space, complete the following steps to resolve the problem.

Analytics offload issues on OpenShift and Cloud Pak for Integration

If your Analytics offload fails, you can perform checks to help determine the cause. Common causes for an offload failure are connection or security-related issues.

When you save the configuration files, their syntax is first validated to verify that it can be applied. However, this does not confirm that other parts of your analytics collection ecosystem are trouble-free, for example, that your endpoints are running, or that the TLS certificates are validated. Errors in these areas may also result in no data output.

To verify issues you will need to examine the logs for one of your Kubernetes analytics ingestion pods. (Note that it may take a few minutes for the pods to pick up changes in the configuration files after edits are applied.)

For information on obtaining logs, refer to [Logging](#).

Following are some common checks you can make to troubleshoot the offload:

- Verify that the specified URL or the host and port values for the target system are accurate.
- Verify that the server or cluster of servers in the target system are running and reachable. If applicable, also verify that the connection problems are not caused by firewall settings.
- If you are using Transport Layer Security (TLS) to establish a private and secure communication channel to the target system, investigate whether the issue is caused by SSL certificate errors; for example, untrusted server certificates, certificates that have expired or are not yet valid, or missing intermediate (or chain) certificates. For more information on using TLS, see [Creating a TLS profile](#).
- If offloading to an Elasticsearch or Kafka target, verify that your configuration settings are valid; for example, the user credentials for authenticating to Elasticsearch or the Kafka topic name.

Empty Analytics display on OpenShift and Cloud Pak for Integration

If your API Connect analytics data doesn't display in the Analytics dashboard, complete the following steps to determine the cause of the problem.

Before you begin

This operation requires the use of the Analytics CLI. [Using the Analytics CLI](#).

Procedure

1. Run the following command to check the logs of the mtlis-gw pod:

```
oc logs analyt-mtlis-gw-7665f5d445-4cx4n -n <APIC_namespace>
```

The response looks like the following example:

```
10.254.6.1 - - [22/Sep/2021:18:05:54 +0000] "POST /x2020/v1/apievent/_bulk HTTP/1.1" 200 2 "-" "-" 2841 0.072 0.070
```

- a. Check the response for any `_bulk` request coming from the gateway, as shown in the example.
If there are no `_bulk` requests, then the problem is with the gateway and the wsm-agent, which is not sending bulk requests to the Analytics server.
- b. If you see the `_bulk` request., then check the return code in the response.
If `_bulk` requests are processed successfully you will see a return code 200 (as in the example), which indicates that the bulk request has been ingested.
- c. If it is any other return code, check the ingestion pod logs for any errors, which will indicate any issues during ingestion:

```
oc logs analyt-ingestion-78d7cff7f7-6pktd
```

2. If everything with the data flow looks fine, and if you are unable to see analytics data, then check the health of Analytics by executing the following CLI command:

```
apic --mode analytics clustermgmt:getHealth --server <management server platform api endpoint> --analytics-service <analytics service name> --format json
```

where:

- `<management server platform api endpoint>` is the Management server admin endpoint.
- `<analytics service name>` is the name of Analytics service as configured in the Cloud Manager.

When you check the cluster's health, the response for a single node analytics deployment looks like the following example:

```
{
  "cluster_name": "apic-analytics-cluster",
  "status": "yellow",
  "timed_out": false,
  "number_of_nodes": 1,
  "number_of_data_nodes": 1,
  "discovered_master": true,
  "discovered_cluster_manager": true,
  "active_primary_shards": 8,
  "active_shards": 8,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 12,
  "delayed_unassigned_shards": 0,
  "number_of_pending_tasks": 0,
  "number_of_in_flight_fetch": 0,
  "task_max_waiting_in_queue_millis": 0,
  "active_shards_percent_as_number": 40
}
```

Note: Single node analytics clusters never show `green` in the health output. Green indicates that the data is replicated to at least one other node, and since there are no other nodes in a single node cluster it remains as `yellow`.

- a. Check the status of the cluster (red, yellow, green) in the response.
In the example, the status is set to yellow.
 - Green indicates all indices are fine.
 - Yellow indicates there are some unassigned shards, which might still be replicating.
 - Red indicates there is a problem with one or more of the indices.
- b. Run the following command to get a list of indices:

```
apic --mode analytics clustermgmt:catIndices --server [SERVER] --analytics-service [ANALYTICS SERVICE]
```

where:

- `SERVER` is the Management server admin endpoint.
- `ANALYTICS SERVICE` is the name of Analytics service as configured in the Cloud Manager.

- c. From the list of indices returned, identify the those with status of 'red'.
- d. Delete each red index by running the following command:

```
apic --mode analytics clustermgmt:deleteIndex --index <index_name> --server [SERVER] --analytics-service [ANALYTICS SERVICE] --format json
```

where:

- `SERVER` is the Management server admin endpoint.
- `ANALYTICS SERVICE` is the name of Analytics service as configured in the Cloud Manager.

Analytics running out of disk space on OpenShift and Cloud Pak for Integration

If your Analytics data is taking up too much disk space, complete the following steps to resolve the problem.

Before you begin

This operation requires the use of the Analytics CLI. [Using the Analytics CLI](#).

About this task

Analytics indices taking up too much space

Procedure

1. Run the following command to display the indices:

```
apic --mode analytics clustermgmt:catIndices --server [SERVER] --analytics-service [ANALYTICS SERVICE]
```

where:

- **SERVER** is the Management server admin endpoint.
- **ANALYTICS SERVICE** is the name of Analytics service as configured in the Cloud Manager.

2. From the output of above command, identify the old and large indices that can be deleted, and delete each index using the following command:

```
apic --mode analytics clustermgmt:deleteIndex --index <index_name> --server [SERVER] --analytics-service [ANALYTICS SERVICE] --format json
```

where:

- **SERVER** is the Management server admin endpoint.
- **ANALYTICS SERVICE** is the name of Analytics service as configured in the Cloud Manager.

3. Reduce the retention period: [Configuring data retention and index rollover time periods](#)
4. Apply a filter to reduce the index size is by excluding large fields, such as **response_body**, from being saved in the index.

For example:

```
filter {
  if "apicapievent" in [tags] {
    mutate {
      remove_field => ["response_body"]
    }
  }
}
```

For information on creating filters for Analytics data, see [Modifying incoming analytics data](#).

Uninstalling API Connect

You can uninstall a Kubernetes deployment of API Connect.

About this task

You can choose to uninstall individual API Connect subsystems or you can uninstall the complete API Connect deployment. You can delete Custom Resource Definitions, Persistent Volumes, service accounts, secrets, operators and the namespace. Use one of the following topics.

Note: For information about how to maintain a two data center deployment, see [Maintaining a two data center deployment](#).

- [Uninstalling API Connect subsystems](#)
You can individually uninstall API Connect subsystems that have been deployed on Kubernetes.
- [Removing API Connect](#)
You can remove all parts of the API Connect deployment on Kubernetes.
- [Removing API Connect on OpenShift or Cloud Pak for Integration](#)
Remove your entire API Connect deployment on OpenShift or Cloud Pak for Integration.

Uninstalling API Connect subsystems

You can individually uninstall API Connect subsystems that have been deployed on Kubernetes.

Before you begin

If you want to disable the Analytics subsystem without uninstalling it, see [Disabling and re-enabling the Analytics subsystem on Kubernetes](#).

About this task

- To delete the API Connect deployment in a Kubernetes runtime environment, you must use the Kubernetes command-line tool, [kubectl](#).
- The API Connect operator and DataPower operator pods are needed to uninstall subsystems individually.
- If need to retain data to re-use during a re-installation later, follow the backup and restore procedures:
 - [Configuring backup settings for fresh install of the Management subsystem](#)
 - [Backing up and restoring the Developer Portal in a Kubernetes environment](#)
- If you want to delete **all** of API Connect, do not use this topic. Instead, see [Removing API Connect](#).

Procedure

1. Uninstall the Management subsystem
 - a. Save the management subsystem name:

```
kubectl get mgmt -n <namespace> -o yaml
```

This command returns the YAML format of the management custom resource. Make a note of the name.

- b. Delete the Management Subsystem.

```
kubectl delete mgmt --all -n <namespace>
```

Important:

If your deployment uses default namespaces be cautious when using `kubectl delete <entity> --all` commands. There can be other pods, services, and more in the default namespace. When using the default namespace, keep in mind that using `kubernetes delete <entity> --all`, where `<entity>` can delete non-APIC entities, such as `sa` (Service Accounts), `secrets`, `rolebindings`, etc.

The Management CR deletion will take some time because the apiconnect operator must delete the Postgres cluster.

- c. Manually delete secrets and PVCs:

- i. Get all of the secrets that use the management subsystem name (which you determined in Step [1.a](#)):

```
kubectl get secrets -n <namespace> | grep <management-subsystem-name>
```

- ii. For each secret that uses the management subsystem name, run the following command to delete the secret:

```
kubectl delete secrets <secret_name> -n <namespace>
```

- iii. Get all the PVCs that use the management subsystem name:

```
kubectl get pvc -n <namespace> | grep <management-subsystem-name>
```

- iv. For each PVC that uses the management subsystem name, run the following command to delete the PVC:

```
kubectl delete pvc <pvc_name> -n <namespace>
```

- d. If management subsystem backup and restore CRs are left stale, delete them also:

- i. Get the name of the backup CR:

```
kubectl get mgmtb -n <namespace>
```

- ii. Delete the backup CR:

```
kubectl delete mgmtb <backup_cr_name> -n <namespace>
```

- iii. Get the name of the restore CR:

```
kubectl get mgmtr -n <namespace>
```

- iv. Delete the restore CR:

```
kubectl delete mgmtr <restore_cr_name> -n <namespace>
```

2. Uninstall Analytics subsystem

- a. Delete the Analytics subsystem, the Analytics backup, and the Analytics restore CRs.

```
kubectl delete a7s --all -n <namespace>
kubectl delete a7b --all -n <namespace>
kubectl delete a7r --all -n <namespace>
```

- b. Manually delete the Analytics secrets:

```
kubectl delete secrets <secret_name> -n <namespace>
```

for each of the following secrets:

```
analytics-ai-endpoint
analytics-ca
analytics-client
analytics-server
```

- c. Manually delete Analytics Persistent Volumes (PVCs):

```
kubectl delete pvc <pvc_name> -n <namespace>
```

for PVCs with the following naming pattern:

```
data-analytics-storage
```

3. Uninstall the Developer Portal subsystem

- a. Delete the Developer Portal subsystem, and the Portal backup and restore CRs.

```
kubectl delete ptl --all -n <namespace>
kubectl delete pb --all -n <namespace>
kubectl delete pr --all -n <namespace>
```

- b. Manually delete Developer Portal secrets:

```
kubectl delete secrets <secret_name> -n <namespace>
```

for secrets with the following naming pattern:

```
portal-admin
portal-admin-client
portal-ca
portal-client
portal-db-ca
```

```
portal-encryption-key-...
portal-server
portal-web
```

c. Manually delete these Developer Portal PVCs:

```
kubectl delete pvc <pvc_name> -n <namespace>
```

for PVCs with the following naming pattern:

```
admin-portal-...-www-...
backup-portal-...-www-...
db-portal-...-db-...
dblogs-portal-...-db-...
web-portal-...-www-...
```

4. Delete the DataPower Gateway subsystem.

```
kubectl delete gw --all -n <namespace>
```

Note: The following entries include both Multi-Protocol Gateway Service (v5-compatible) and API Gateway settings. Names that include "gwv5" refer to the Multi-Protocol Gateway Service (v5-compatible), and names that include "gwv6" refer to the API Gateway.

a. Delete Service Accounts.

```
kubectl delete sa <service_account> -n <namespace>
```

for each of the following Service Accounts:

```
gwv5-datapower
gwv6-datapower
```

b. Delete Secrets.

```
kubectl delete secrets <secret_name> -n <namespace>
```

for secrets with the following naming pattern:

```
gwv5-datapower-token-...
gwv5-dpm-sa-token-...
gwv5-endpoint
gwv5-manager-endpoint
gwv6-datapower-token-...
gwv6-dpm-sa-token-...
gwv6-endpoint.
gwv6-manager-endpoint
```

c. Delete PVCs.

```
kubectl delete pvc <pvc_name> -n <namespace>
```

for PVCs with the following naming pattern:

```
gwv5...
gwv6...
```

- [Removing Management subsystem postgres resources on Kubernetes](#)

When you remove the API Connect Management subsystem by deleting its files, some postgres resources from Crunchy Data remain, and must be deleted manually before you attempt to re-install the Management subsystem.

Removing Management subsystem postgres resources on Kubernetes

When you remove the API Connect Management subsystem by deleting its files, some postgres resources from Crunchy Data remain, and must be deleted manually before you attempt to re-install the Management subsystem.

About this task

Sometimes when you remove the Management subsystem, some postgres resources remain and must be deleted manually.

To determine whether you need to delete any postgres resources, use the `kubectl get` command and specify `vendor=crunchydata`; as shown in the following example.

```
$ kubectl get pods -n apic -l vendor=crunchydata
NAME                                READY   STATUS    RESTARTS   AGE
backrest-backup-production-mgmt-mymgmtsite123-postgres-k77xj  0/1     Completed 0          17h
production-mgmt-mymgmtsite123-postgres-backrest-shared-repk4kzc 1/1     Running   0          2d21h
production-mgmt-mymgmtsite123-postgres-full-sch-backup-9ktp5    0/1     Completed 0          164m

$ kubectl get deployments -n apic -l vendor=crunchydata
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
production-mgmt-mymgmtsite123-postgres-backrest-shared-repo    1/1     1             1          2d21h

$ kubectl get sa -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ kubectl get jobs -n apic -l vendor=crunchydata
NAME                                COMPLETIONS   DURATION   AGE
backrest-backup-production-mgmt-mymgmtsite123-postgres         1/1           10s       17h
production-mgmt-mymgmtsite123-postgres-full-sch-backup         1/1           37s       164m
production-mgmt-mymgmtsite123-postgres-rmdata-cemg             1/1           20s       14m
```

```

$ kubectl get roles -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ kubectl get rolebindings -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ kubectl get secrets -n apic -l vendor=crunchydata
NAME                                     TYPE      DATA      AGE
production-mgmt-7bc754c7-postgres-backrest-repo-config  Opaque    9          6d23h
production-mgmt-7bc754c7-postgres-postgres-secret      Opaque    2          6d23h
production-mgmt-7bc754c7-postgres-primaryuser-secret   Opaque    2          6d23h
production-mgmt-7bc754c7-postgres-testuser-secret      Opaque    2          6d23h
production-mgmt-8aac6b4e-postgres-backrest-repo-config  Opaque    9          6d22h
production-mgmt-8aac6b4e-postgres-postgres-secret      Opaque    2          6d22h
production-mgmt-8aac6b4e-postgres-primaryuser-secret   Opaque    2          6d22h
production-mgmt-8aac6b4e-postgres-testuser-secret      Opaque    2          6d22h
production-mgmt-mymgmtsite123-postgres-backrest-repo-config  Opaque    9          2d21h
production-mgmt-mymgmtsite123-postgres-postgres-secret   Opaque    2          2d21h
production-mgmt-mymgmtsite123-postgres-primaryuser-secret Opaque    2          2d21h
production-mgmt-mymgmtsite123-postgres-testuser-secret   Opaque    2          2d21h

$ kubectl get cm -n apic -l vendor=crunchydata
NAME                                     DATA      AGE
production-mgmt-mymgmtsite123-postgres-config           0          2d21h
production-mgmt-mymgmtsite123-postgres-leader          0          2d21h
production-mgmt-mymgmtsite123-postgres-pgbouncer-cm    2          2d21h
production-mgmt-mymgmtsite123-postgres-pgha-config     5          2d21h

```

Procedure

For each type of postgres resource, check to see if any resources are available and if so, delete the resources:

- pods:


```

kubectl get pods -n <namespace> -l vendor=crunchydata
kubectl delete pods -n <namespace> -l vendor=crunchydata

```
- deployments:


```

kubectl get deployments -n <namespace> -l vendor=crunchydata
kubectl delete deployments -n <namespace> -l vendor=crunchydata

```
- service accounts:


```

kubectl get sa -n <namespace> -l vendor=crunchydata
kubectl delete sa -n <namespace> -l vendor=crunchydata

```
- jobs:


```

kubectl get jobs -n <namespace> -l vendor=crunchydata
kubectl delete jobs -n <namespace> -l vendor=crunchydata

```
- roles:


```

kubectl get roles -n <namespace> -l vendor=crunchydata
kubectl delete roles -n <namespace> -l vendor=crunchydata

```
- rolebindings:


```

kubectl get rolebindings -n <namespace> -l vendor=crunchydata
kubectl delete rolebindings -n <namespace> -l vendor=crunchydata

```
- secrets:


```

kubectl get secrets -n <namespace> -l vendor=crunchydata
kubectl delete secrets -n <namespace> -l vendor=crunchydata

```
- cm:


```

kubectl get cm -n <namespace> -l vendor=crunchydata
kubectl delete cm -n <namespace> -l vendor=crunchydata

```
- pgtasks:


```

kubectl get pgtasks.crunchydata.com -n <namespace> -l vendor=crunchydata
kubectl delete pgtasks.crunchydata.com -n <namespace>

```
- pgreplica:


```

kubectl get pgreplica -n <management_namespace> -l vendor=crunchydata
kubectl delete pgreplica -n <management_namespace> -l vendor=crunchydata

```

Removing API Connect

You can remove all parts of the API Connect deployment on Kubernetes.

About this task

These instructions completely remove API Connect, including PVCs and secrets.

- To delete the API Connect deployment in a Kubernetes runtime environment, you must use the Kubernetes command-line tool, [kubect1](#).
- You must run the commands in the order shown in the procedure on this page. If you run commands out of order, you might encounter a problem where the deletion of namespaces hangs.
- If you want to retain the PVCs, which contain some of the data you built up when using the application, you must also keep the associated secret used to decrypt the data. To do this, you must backup each subsystem, and then uninstall the subsystems individually. In this case, do not use the instructions on this page. Instead, see [Uninstalling API Connect subsystems](#).

Procedure

1. Delete all API Connect systems.

```
kubect1 delete apic --all -n <namespace>
```

This command is the equivalent of deleting each of the individual subsystems one at a time, and also deleting all of their backup and restore CRs. The commands that run during this step are:

- Subsystem CRs. For example:

```
kubect1 delete mgmt --all -n <namespace>
kubect1 delete a7s --all -n <namespace>
kubect1 delete pt1 --all -n <namespace>
kubect1 delete gwv6 --all -n <namespace>
```

- Backup and restore CRs:

```
kubect1 delete mgmtb --all -n <namespace>
kubect1 delete mgmtr --all -n <namespace>
kubect1 delete pb --all -n <namespace>
kubect1 delete pr --all -n <namespace>
kubect1 delete a7b --all -n <namespace>
kubect1 delete a7r --all -n <namespace>
```

2. Delete the Operator deployments.

```
kubect1 delete deployment datapower-operator ibm-apiconnect -n <namespace>
```

3. Delete all Service Accounts in the namespace.

Important:

If your deployment uses default namespaces be cautious when using `kubect1 delete`

`<entity> --all` commands. There can be other pods, services, and more in the default namespace. When using the default namespace, keep in mind that using `kubernetes delete <entity> --all`, where `<entity>` can delete non-API entities, such as `sa` (Service Accounts), `secrets`, `rolebindings`, etc.

```
kubect1 delete sa --all -n <namespace>
```

4. Delete all Secrets in the namespace.

```
kubect1 delete secrets --all -n <namespace>
```

5. Delete all Rolebindings in the namespace.

```
kubect1 delete rolebindings --all -n <namespace>
```

6. Delete all Roles in the namespace.

```
kubect1 delete roles --all -n <namespace>
```

7. Clean up any remaining jobs.

```
kubect1 delete jobs --all -n <namespace>
```

8. Delete all PVCs in the namespace.

```
kubect1 delete pvc --all -n <namespace>
```

9. Delete Issuers in the namespace.

Note that there might be none found.

```
kubect1 delete issuer --all -n <namespace>
```

10. Delete certificates in the namespace.

Note that there might be none found.

```
kubect1 delete certificates --all -n <namespace>
```

11. If you are not using the default namespace, delete the Namespace

Important: If your deployment uses the default namespace, do not delete it. There can be other pods, services, and more in the default namespace.

```
kubect1 delete ns <namespace>
```

12. Delete validatingwebhookconfiguration

- a. `kubectl get validatingwebhookconfiguration | awk /<ns>/'{print $1}' | xargs kubectl delete validatingwebhookconfiguration`
- b. `kubectl delete validatingwebhookconfiguration managements.validator.subsystem.apiconnect.ibm.com`

13. Delete mutatingwebhookconfiguration

- a. `kubectl get mutatingwebhookconfiguration | awk /$NS/'{print $1}' | xargs kubectl delete mutatingwebhookconfiguration`
- b. `kubectl get mutatingwebhookconfiguration | awk /apicop-management-defaulter/'{print $1}' | xargs kubectl delete mutatingwebhookconfiguration`

14. Delete the Apiconnect CRDs (custom resource definitions)

```
kubectl delete crd analyticsbackups.analytics.apiconnect.ibm.com
kubectl delete crd analyticsclusters.analytics.apiconnect.ibm.com
kubectl delete crd analyticsrestores.analytics.apiconnect.ibm.com
kubectl delete crd datapowerservices.datapower.ibm.com
kubectl delete crd gatewayclusters.gateway.apiconnect.ibm.com
kubectl delete crd managementbackups.management.apiconnect.ibm.com
kubectl delete crd managementclusters.management.apiconnect.ibm.com
kubectl delete crd managementrestores.management.apiconnect.ibm.com
kubectl delete crd natsclusters.nats.io
kubectl delete crd natsserviceroles.nats.io
kubectl delete crd natsstreamingclusters.streaming.nats.io
kubectl delete crd portalbackups.portal.apiconnect.ibm.com
kubectl delete crd portalclusters.portal.apiconnect.ibm.com
kubectl delete crd portalrestores.portal.apiconnect.ibm.com
```

15. Delete Crunchy CRDs

```
kubectl delete crd pgclusters.crunchydata.com
kubectl delete crd pgpolicies.crunchydata.com
kubectl delete crd pgreplicas.crunchydata.com
kubectl delete crd pgtasks.crunchydata.com
```

16. If you installed Cert-manager and want to remove it now, complete the following steps:

Warning: If you are using shared clusters for API Connect with a common cert manager, be aware that this step removes the components of the cert manager from cluster wide scope.

a. Delete the Cert-manager

```
kubectl delete deploy --all -n cert-manager
kubectl delete service --all -n cert-manager
kubectl delete sa -n cert-manager --all
kubectl delete secrets -n cert-manager --all
kubectl delete APIService v1beta1.webhook.cert-manager.io -n cert-manager
kubectl delete MutatingWebhookConfiguration cert-manager-webhook -n cert-manager
kubectl delete ValidatingWebhookConfiguration cert-manager-webhook -n cert-manager
kubectl delete ns cert-manager
```

b. Delete cert-manager CRDs

- Delete the cert-manager CRDs with the following:

```
kubectl delete crd issuers.cert-manager.io
kubectl delete crd certificaterequests.cert-manager.io
kubectl delete crd certificates.cert-manager.io
kubectl delete crd challenges.cert-manager.io
kubectl delete crd clusterissuers.cert-manager.io
kubectl delete crd orders.cert-manager.io
```

- Older API Connects versions may have cert-manager CRDs from an older version of cert-manager which had a different name, delete these CRDs with the following:

```
kubectl delete crd certificaterequests.certmanager.k8s.io
kubectl delete crd certificates.certmanager.k8s.io
kubectl delete crd challenges.acme.certmanager.k8s.io
kubectl delete crd clusterissuers.certmanager.k8s.io
kubectl delete crd issuers.certmanager.k8s.io
kubectl delete crd orders.acme.certmanager.k8s.io
```

- [Removing postgres resources on Kubernetes after removing API Connect](#)

When you remove API Connect by deleting its files, some of the postgres resources from Crunchy Data remain and must be deleted manually before you attempt to re-install API Connect.

Removing postgres resources on Kubernetes after removing API Connect

When you remove API Connect by deleting its files, some of the postgres resources from Crunchy Data remain and must be deleted manually before you attempt to re-install API Connect.

About this task

Sometimes when you remove API Connect, some postgres resources remain and must be deleted manually.

To determine whether you need to delete any postgres resources, use the `kubect1` `get` command and specify `vendor=crunchydata` as shown in the following example.

```
$ kubect1 get pods -n apic -l vendor=crunchydata
NAME                                READY   STATUS    RESTARTS   AGE
backrest-backup-production-mgmt-mymgmtsite123-postgres-k77xj    0/1     Completed 0           17h
production-mgmt-mymgmtsite123-postgres-backrest-shared-repk4kzc  1/1     Running   0           2d21h
production-mgmt-mymgmtsite123-postgres-full-sch-backup-9ktp5     0/1     Completed 0           164m

$ kubect1 get deployments -n apic -l vendor=crunchydata
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
production-mgmt-mymgmtsite123-postgres-backrest-shared-repo     1/1     1             1           2d21h

$ kubect1 get sa -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ kubect1 get jobs -n apic -l vendor=crunchydata
NAME                                COMPLETIONS   DURATION   AGE
backrest-backup-production-mgmt-mymgmtsite123-postgres          1/1           10s        17h
production-mgmt-mymgmtsite123-postgres-full-sch-backup          1/1           37s        164m
production-mgmt-mymgmtsite123-postgres-rmdata-cemg              1/1           20s        14m

$ kubect1 get roles -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ kubect1 get rolebindings -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ kubect1 get secrets -n apic -l vendor=crunchydata
NAME                                TYPE   DATA   AGE
production-mgmt-7bc754c7-postgres-backrest-repo-config          Opaque 9        6d23h
production-mgmt-7bc754c7-postgres-postgres-secret              Opaque 2        6d23h
production-mgmt-7bc754c7-postgres-primaryuser-secret           Opaque 2        6d23h
production-mgmt-7bc754c7-postgres-testuser-secret              Opaque 2        6d23h
production-mgmt-8aac6b4e-postgres-backrest-repo-config          Opaque 9        6d22h
production-mgmt-8aac6b4e-postgres-postgres-secret              Opaque 2        6d22h
production-mgmt-8aac6b4e-postgres-primaryuser-secret           Opaque 2        6d22h
production-mgmt-8aac6b4e-postgres-testuser-secret              Opaque 2        6d22h
production-mgmt-mymgmtsite123-postgres-backrest-repo-config     Opaque 9        2d21h
production-mgmt-mymgmtsite123-postgres-postgres-secret          Opaque 2        2d21h
production-mgmt-mymgmtsite123-postgres-primaryuser-secret       Opaque 2        2d21h
production-mgmt-mymgmtsite123-postgres-testuser-secret          Opaque 2        2d21h

$ kubect1 get cm -n apic -l vendor=crunchydata
NAME                                DATA   AGE
production-mgmt-mymgmtsite123-postgres-config                    0       2d21h
production-mgmt-mymgmtsite123-postgres-leader                    0       2d21h
production-mgmt-mymgmtsite123-postgres-pgbouncer-cm              2       2d21h
production-mgmt-mymgmtsite123-postgres-pgha-config                5       2d21h
```

Procedure

For each type of postgres resources, check to see if any resources are available and if so, delete the resources:

- pods:

```
kubect1 get pods -n <namespace> -l vendor=crunchydata
kubect1 delete pods -n <namespace> -l vendor=crunchydata
```
- deployments:

```
kubect1 get deployments -n <namespace> -l vendor=crunchydata
kubect1 delete deployments -n <namespace> -l vendor=crunchydata
```
- Jobs:

```
kubect1 get jobs -n <namespace> -l vendor=crunchydata
kubect1 delete jobs -n <namespace> -l vendor=crunchydata
```
- roles:

```
kubect1 get roles -n <namespace> -l vendor=crunchydata
kubect1 delete roles -n <namespace> -l vendor=crunchydata
```
- rolebindings:

```
kubect1 get rolebindings -n <namespace> -l vendor=crunchydata
kubect1 delete rolebindings -n <namespace> -l vendor=crunchydata
```
- secrets:

```
kubect1 get secrets -n <namespace> -l vendor=crunchydata
kubect1 delete secrets -n <namespace> -l vendor=crunchydata
```
- cm:

```
kubect1 get cm -n <namespace> -l vendor=crunchydata
kubect1 delete cm -n <namespace> -l vendor=crunchydata
```


- pgtasks:


```
kubectl get pgtasks.crunchydata.com -n <namespace> -l vendor=crunchydata
kubectl delete pgtasks.crunchydata.com -n <namespace>
```
- pgreplica:


```
kubectl get pgreplica -n <management_namespace> -l vendor=crunchydata
kubectl delete pgreplica -n <management_namespace> -l vendor=crunchydata
```

Removing API Connect on OpenShift or Cloud Pak for Integration

Remove your entire API Connect deployment on OpenShift or Cloud Pak for Integration.

About this task

These instructions completely remove API Connect, including PVCs and secrets.

Attention: Run the commands in the sequence shown on this page. If you run commands out of sequence, you might encounter a problem where the deletion of namespaces hangs.

Procedure

1. If Event Endpoint Management is installed, run the following command to determine the name of your Event Endpoint Management instance:

```
oc get eem -n <APIC-namespace>
```

2. If you are using a top-level CR, run the following command to determine the name of your API Connect instance:

```
oc get apiconnectcluster -n <APIC-namespace>
```

3. If you are using a top-level CR, run the following command to delete the API Connect operands (subsystems):

```
oc delete apiconnectcluster <name_of_apic_instance> -n <APIC-namespace>
```

4. Run the following command to delete the API Connect backup and restore CRs:

```
oc delete apic --all -n <APIC-namespace>
```

5. Run the following command to delete the secrets that are associated with your deleted API Connect cluster:

```
oc get secrets --no-headers -n <APIC-namespace> | cut -d' ' -f1 | grep "^<first_10_characters_of_apic_instance_name>-" | xargs oc delete secret -n <APIC-namespace>
```

Where `<first_10_characters_of_apic_instance_name>` refers to the name of your API Connect instance.

Note: This step assumes that any secrets that you want to keep in your namespace do not have a name that starts with the same 10 first characters of the API Connect Cluster name.

6. If Event Endpoint Management is installed, run the following command to delete the secrets that are associated with your deleted Event Endpoint Management instance:

```
oc get secrets --no-headers -n <APIC-namespace> | cut -d' ' -f1 | grep "^<first_10_characters_of_eem_instance_name>-" | xargs oc delete secret -n <APIC-namespace>
```

Where `<first_10_characters_of_eem_instance_name>` refers to the name of your Event Endpoint Manager instance.

Note: This step assumes that any secrets that you want to keep in your namespace do not have a name that starts with the same 10 first characters of the Event Endpoint Manager name.

7. If the only software that is deployed in your namespace is API Connect, run the following command to delete the PVCs associated with your deleted API Connect cluster:

```
oc delete pvc -n <APIC-namespace> -l app.kubernetes.io/managed-by=ibm-apiconnect
```

8. If you have no other software in your namespace that uses a CrunchyData Postgres cluster, delete the CrunchyData Postgres cluster:

```
oc delete pvc -n <APIC-namespace> -l vendor=crunchydata
```

9. Check for and remove any other Crunchy data: [Removing postgres resources on OpenShift and Cloud Pak for Integration](#).

10. Run the following commands to delete the API Connect secret rotation CRs for the cluster:

```
oc delete mgmtsr -n <APIC-namespace> --all
oc delete ptlsr -n <APIC-namespace> --all
```

11. Optional: If you created a namespace specifically for API Connect, and are not using it for anything else, you can delete the namespace. If your API Connect operators are installed in this namespace, instead of in a shared global namespace, they are also deleted.

```
oc delete ns <APIC-namespace>
```

Deleting the namespace removes any remaining certificates and secrets.

Note: Namespace deletion can get stuck, check that the namespace is deleted with:

```
oc get ns <APIC-namespace>
```

If the namespace is still present, to see why deletion is stuck, run:

```
oc describe ns <APIC-namespace>
```

12. Optional: Delete the API Connect operator deployments.

If your API Connect operator deployments are in a global namespace they are not deleted when your API Connect namespace is deleted. To delete the API Connect operators:

- Open the OpenShift web console and go to Installed Operators, in the namespace where the API Connect operators are installed.
- For each operator, click the options menu, and select Uninstall Operator.

API Connect operator might have created a number of operand requests with IBM Cloud Pak foundational services (previously called Common Services) to deploy services such as Cert-Manager. In addition, API Connect requires the use of a DataPower Gateway, so the DataPower operator might be installed in the API Connect namespace. If those artifacts are no longer needed, uninstall them individually.

Note: DataPower can be deployed on its own without API Connect. It is possible that the DataPower operator was installed for another purpose and not for use with API Connect, so check with your organization before you delete it.

Deleting the operators from the web console ensures that related subscriptions, CSVs, and install plans are also removed.

Removing postgres resources on OpenShift and Cloud Pak for Integration

When you remove API Connect by deleting its files, some of the postgres resources from Crunchy Data remain and must be deleted manually before you attempt to re-install API Connect.

About this task

Sometimes when you remove API Connect, some postgres resources remain and must be deleted manually.

To determine whether you need to delete any postgres resources, use the `oc get` command and specify `vendor=crunchydata` as shown in the following example.

```
$ oc get pods -n apic -l vendor=crunchydata
NAME                                     READY   STATUS    RESTARTS   AGE
backrest-backup-production-mgmt-mymgmtsite123-postgres-k77xj   0/1     Completed 0           17h
production-mgmt-mymgmtsite123-postgres-backrest-shared-repk4kzc 1/1     Running   0           2d21h
production-mgmt-mymgmtsite123-postgres-full-sch-backup-9ktp5    0/1     Completed 0           164m

$ oc get deployments -n apic -l vendor=crunchydata
NAME                                     READY   UP-TO-DATE   AVAILABLE   AGE
production-mgmt-mymgmtsite123-postgres-backrest-shared-repo    1/1     1             1           2d21h

$ oc get sa -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ oc get jobs -n apic -l vendor=crunchydata
NAME                                     COMPLETIONS   DURATION   AGE
backrest-backup-production-mgmt-mymgmtsite123-postgres         1/1           10s        17h
production-mgmt-mymgmtsite123-postgres-full-sch-backup         1/1           37s        164m
production-mgmt-mymgmtsite123-postgres-rmdata-cemg             1/1           20s        14m

$ oc get roles -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ oc get rolebindings -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ oc get secrets -n apic -l vendor=crunchydata
NAME                                     TYPE   DATA   AGE
production-mgmt-7bc754c7-postgres-backrest-repo-config         Opaque 9        6d23h
production-mgmt-7bc754c7-postgres-postgres-secret              Opaque 2        6d23h
production-mgmt-7bc754c7-postgres-primaryuser-secret          Opaque 2        6d23h
production-mgmt-7bc754c7-postgres-testuser-secret              Opaque 2        6d23h
production-mgmt-8aac6b4e-postgres-backrest-repo-config         Opaque 9        6d22h
production-mgmt-8aac6b4e-postgres-postgres-secret              Opaque 2        6d22h
production-mgmt-8aac6b4e-postgres-primaryuser-secret          Opaque 2        6d22h
production-mgmt-8aac6b4e-postgres-testuser-secret              Opaque 2        6d22h
production-mgmt-mymgmtsite123-postgres-backrest-repo-config    Opaque 9        2d21h
production-mgmt-mymgmtsite123-postgres-postgres-secret         Opaque 2        2d21h
production-mgmt-mymgmtsite123-postgres-primaryuser-secret      Opaque 2        2d21h
production-mgmt-mymgmtsite123-postgres-testuser-secret         Opaque 2        2d21h

$ oc get cm -n apic -l vendor=crunchydata
NAME                                     DATA   AGE
production-mgmt-mymgmtsite123-postgres-config                   0       2d21h
production-mgmt-mymgmtsite123-postgres-leader                   0       2d21h
production-mgmt-mymgmtsite123-postgres-pgbouncer-cm            2       2d21h
production-mgmt-mymgmtsite123-postgres-pgha-config              5       2d21h
```

Procedure

For each type of postgres resource, check to see if any resources are available and if so, delete the resources:

- Pods:

```
oc get pods -n <APIC_namespace> -l vendor=crunchydata
oc delete pods -n <APIC_namespace> -l vendor=crunchydata
```

- Deployments:

```
oc get deployments -n <APIC_namespace> -l vendor=crunchydata
```

```
oc delete deployments -n <APIC_namespace> -l vendor=crunchydata
```

- Jobs:

```
oc get jobs -n <APIC_namespace> -l vendor=crunchydata
```

```
oc delete jobs -n <APIC_namespace> -l vendor=crunchydata
```

- roles:

```
oc get roles -n <APIC_namespace> -l vendor=crunchydata
```

```
oc delete roles -n <APIC_namespace> -l vendor=crunchydata
```

- rolebindings:

```
oc get rolebindings -n <APIC_namespace> -l vendor=crunchydata
```

```
oc delete rolebindings -n <APIC_namespace> -l vendor=crunchydata
```

- secrets:

```
oc get secrets -n <APIC_namespace> -l vendor=crunchydata
```

```
oc delete secrets -n <APIC_namespace> -l vendor=crunchydata
```

- cm:

```
oc get cm -n <APIC_namespace> -l vendor=crunchydata
```

```
oc delete cm -n <APIC_namespace> -l vendor=crunchydata
```

- pgtasks

```
oc get pgtasks.crunchydata.com -n <APIC_namespace> -l vendor=crunchydata
```

```
oc delete pgtasks.crunchydata.com -n <APIC_namespace>
```

- pgreplica:

```
oc get pgreplica -n <APIC_namespace> -l vendor=crunchydata
```

```
oc delete pgreplica -n <APIC_namespace> -l vendor=crunchydata
```

VMware

Use the instructions in this section to install, upgrade, and maintain API Connect on VMware.

Attention:

Although API Connect 10.0.5.0 is not supported to run on VMware, support for VMware is available from API Connect 10.0.5.1 and later.

- [Planning your deployment](#)
Before you install API Connect, understand deployment options and requirements for endpoints, load balancing, certificates, and firewalls.
- [Installing API Connect](#)
Use these instructions to install or upgrade a deployment of API Connect on VMware.
- [Upgrading API Connect](#)
Upgrade API Connect on VMware to a newer version.
- [Maintaining API Connect](#)
You can use utilities to complete maintenance tasks such as backup, restore, and certificate management in a VMware environment.

Planning your deployment

Before you install API Connect, understand deployment options and requirements for endpoints, load balancing, certificates, and firewalls.

- [Deployment overview for endpoints and certificates](#)
When deploying API Connect, you will create one or more endpoints for the subsystems and then configure certificates or mutual TLS for most endpoints.
- [Planning your deployment topology and profiles](#)
When you install on VMware, API Connect provides a choice of deployment configurations for capacity and high availability (HA). These configurations are named 'deployment profiles' and there are several pre-defined profiles to choose from.
- [Enable JWT instead of mTLS](#)
- [Firewall requirements on VMware](#)
Diagram for port configuration, and list of active ports, for an IBM® API Connect deployment on VMware.
- [Load balancer configuration in a VMware deployment](#)
When deploying API Connect for High Availability, it is recommended that you configure a cluster with at least three nodes and a load balancer. A sample configuration is provided for placing a load balancer in front of your API Connect OVA deployment.
- [A two data center deployment strategy on VMware](#)
An overview of the two data center disaster recovery deployment strategy in API Connect.
- [Planning your Analytics deployment](#)
Plan your IBM API Connect Analytics deployment by reviewing options for topology, data modifications, and storage.
- [Working with certificates](#)
Use the `certs` command included in the APICUP installer to set and manage certificates for each subsystem. Default certificates are automatically applied, but defaults can be overridden by user-supplied custom certificates.

- [Tips and tricks for using APICUP](#)
The APICUP installer contains built-in time saving functions.

Deployment overview for endpoints and certificates

When deploying API Connect, you will create one or more endpoints for the subsystems and then configure certificates or mutual TLS for most endpoints.

Configuring endpoints

Subsystem	Endpoints	Description	Certificates
Management	cloud-admin-ui	Configured using APICUP installer. Endpoint on the management server for communication with the Cloud Manager user interface.	cloud-admin-ui
	api-manager-ui	Configured using APICUP installer. API Manager URL endpoint on the management server for communication with the API Manager user interface.	api-manager-ui
	consumer-api	Configured using APICUP installer. Platform REST API endpoint for running consumer APIs on the management server.	consumer-api
	platform-api	Configured using APICUP installer. Platform REST API endpoint for running admin and provider APIs on the management server.	platform-api
	hub	Automated Testing Behavior UI and API endpoint. External Frontend/Ingress, port 443	hub-endpoint
	turnstile	Automated Testing Behavior UI and API endpoint. External Frontend/Ingress, port 443	turnstile-endpoint
Portal	portal-admin	Configured using APICUP installer. Corresponds to Management Endpoint entered in Cloud Manager. Requires a TLS profile configured with mutual TLS.	mutual TLS
	portal-www	Configured using APICUP installer. Portal Web site URL entered in Cloud Manager. Used publicly to access Portal.	portal-www-ingress
Analytics	analytics-client	This is a legacy certificate, it is not used from v10.0.5 onwards.	mutual TLS
	analytics-ingestion	Configured using APICUP installer. The analytics-ingestion endpoint must be entered in the Cloud Manager UI when registering the analytics service. It is also used by the Gateway service to push data to the Analytics service. Requires a TLS profile configured with mutual TLS.	mutual TLS
Gateway	apic-gw-service	Configured using APICUP installer. This is the endpoint the gateway uses for network communication. Enter this endpoint as the Management Endpoint entered in Cloud Manager.	apic-gw-service-ingress
	api-gateway	Configured using APICUP installer. This is the endpoint the gateway uses for API traffic. Enter this endpoint as the API Invocation Endpoint in Cloud Manager.	api-gateway-ingress

The endpoints are configured by the installation Operator. They are set for each subsystem. Endpoints are also entered when configuring the Topology for the Gateway, Portal, and Analytics subsystems in Cloud Manager.

For instructions on configuring endpoints and installing into an OVA environment, see [Installing API Connect](#).

Configuring certificates

The certificates are configured using the `apicup` command. The certificates for the endpoints are usually configured as custom certificates as described in [Setting custom certificates](#).

Configuring mutual TLS

Mutual TLS is configured for TLS profiles in Cloud Manager. See [Creating a TLS Server Profile](#).

Configuring a proxy

If a Developer Portal is deployed externally to the management server zone, it does not have access to the consumer and product APIs. You need to configure a proxy to enable communication. For more information, see [Configuring a proxy](#).

Planning your deployment topology and profiles

When you install on VMware, API Connect provides a choice of deployment configurations for capacity and high availability (HA). These configurations are named 'deployment profiles' and there are several pre-defined profiles to choose from.

High availability versus disaster recovery

High availability focuses on minimizing the loss of service that is experienced by users during a hardware or software failure. Disaster recovery (DR) focuses on the procedures for recovering a system after a catastrophic hardware, software, or operator failure. Consider the following two metrics:

Recovery Time Objective (RTO)

The RTO is the time that it is acceptable for a system to be unavailable after a disaster.

Recovery Point Objective (RPO)

DR solutions are usually based on restoring from a backup. This means that the system can only be recovered to the state it was in when the last backup was taken, and not to the state it was in at the instant the disaster occurred. The RPO measures how far back in time the recovery point is, and therefore how much new data is lost. An RPO of zero would assert that no data is lost, but such a solution is often a compromise against the cost and performance of the system.

To achieve high availability in your API Connect VMware deployment, the three replica deployment profiles should be used, where each API Connect component is distributed across three virtual machines (VMs). However, this topology requires a low network latency between the VMs that is often unachievable between geographically separated data centers. To overcome this limitation, API Connect provides a two data center deployment solution that has both a low Recovery Time Objective (RTO), and a low Recovery Point Objective (RPO).

Deployment options

API Connect is composed of three components: Management, Portal, and Analytics, and the DataPower Gateway supporting program, see [API Connect components](#). Each of these components includes a set of deployment profiles to cover different capacity and availability requirements.

Note: Throughout this document, the Management, Portal, and Analytics components, and the Gateway supporting program are also referred to as "subsystems". The naming convention of the deployment profiles is **nAx_cB.mC** where:

- **nA** indicates the number (A) of VMs (nodes) that are deployed to support the profile. The term "node" is used to refer to a VM instance (n1 means one node, n3 means three nodes).
- **cB** indicates the minimum number of virtual cores (B) required on the VM to deploy the profile.
- **mC** indicates the minimum amount of memory (C) in GB required by the profile on the VM.

API Connect components have two profile types: one replica and three replica:

- The one replica profile is a single node profile and is also known as the **n1** profile. It consists of a single VM for each component. This deployment type is recommended for development or testing where high availability is not a requirement.
- The three replica profiles consist of a cluster of three VMs for each component. This profile is also referred to as the **n3** profile. Each VM runs a replica of the component's microservices, and if a VM fails, the replicas on the other two VMs continue serving requests.

The available profiles for the Management, Portal, and Analytics components are:

- **Management**
n1xc4.m16, n3xc4.m16.
- **Portal**
n1xc2.m8, n1xc4.m16, n1xc8.m16, n3xc3.m8, n3xc4.m8, n3xc8.m16.
- **Analytics**
n1xc2.m16, n1xc4.m32, n1xc6.m48, n3xc4.m16, n3xc6.m48, n3xc8.m64.

For information about gateway CPU and memory, refer to *Deploying the OVA on a VMware hypervisor* in the appropriate version of the [DataPower documentation](#).

Additional points:

- Deployment profiles do not have to be the same for each component. For example, you can have a three replica deployment of the Management component that is configured with a one replica deployment of the Portal component.
- It is possible to change the deployment profile of an existing installation, but all endpoints in your deployment must remain the same. This means that you should use a load-balancer or logical DNS records when you do the initial one replica deployment. For example, if you installed a one replica deployment and used the portal VM hostname as the portal management registration endpoint, this endpoint cannot change when you move it to a three replica deployment. For more information, see [Changing deployment profiles on VMware](#). If you want to change the deployment profile and the endpoints, you can use the form factor migration procedure: [Migrating from v10 to v10 on a different form factor](#).
- Multiple Portal, Analytics, and Gateway component instances can be added with either configuration. For example, you can have a one replica Management component that is configured with two Portal services: a three replica Portal component, and a one replica Portal component.

Two data center disaster recovery

API Connect deployments can operate across two data centers in an active/warm-standby configuration. Each data center has a complete API Connect installation. The active data center processes all user operations. The warm-standby data center does not process user operations but maintains its management and portal component databases in synchronization with the active data center. If a failure occurs at the active data center, the warm-standby data center can become active with a manual failover operation. For more information on the two data center disaster recovery configuration, see: [A two data center deployment strategy on VMware](#).

Enable JWT instead of mTLS

If your network infrastructure requires that load-balancers implement TLS termination, then mTLS between API Connect subsystems can be disabled and JSON Web Token (JWT) security can be used instead.

Note: Although mTLS is disabled, the network communication is still secured with standard TLS, which does not require passthrough to be enabled on the load-balancers. The following communication flows can be secured with JWT:

- Management initiated communication to portal, analytics, and gateway subsystems. From v10.0.5.3 and later.
- Analytics ingestion: The flow of API event records from the gateway to the analytics subsystem. From v10.0.5.4 and later.
Note: If the gateway is under a high transaction load and sending many API events to the analytics subsystem, the enablement of JWT might impact the performance of your analytics subsystem. It is recommended to run performance tests under your expected analytics load before you decide to use JWT for the analytics ingestion path.

With JWT enabled, the portal, gateway, and analytics subsystems verify the JSON Web Token (JWT) sent from the management subsystem when it initiates communication with them. The analytics subsystem verifies the JWT sent from the gateway with incoming API event data. The subsystems that receive the token contact a JSON Web Key Set (JWKS) URL to verify it. The JWKS URL is hosted by the management subsystem, in a subpath of the management subsystem's platform REST API.

You can configure JWT instead of mTLS during installation, as documented in [Installing API Connect](#), or you can configure JWT after installation: [Use JWT security instead of mTLS between subsystems](#).

If you disable mTLS, you must enable JWT. It is not possible to configure API Connect with both mTLS and JWT disabled.

Firewall requirements on VMware

Diagram for port configuration, and list of active ports, for an IBM® API Connect deployment on VMware.

Required Ports between zones

The following network diagram example helps to explain which ports must be configured in an API Connect network. Specific ports must be configured to enable the communication between the various zones, both public and private, in a network.

The ports specified in the diagram are default ports. Check your deployment to understand which communication, if any, is configured to use non-default ports.

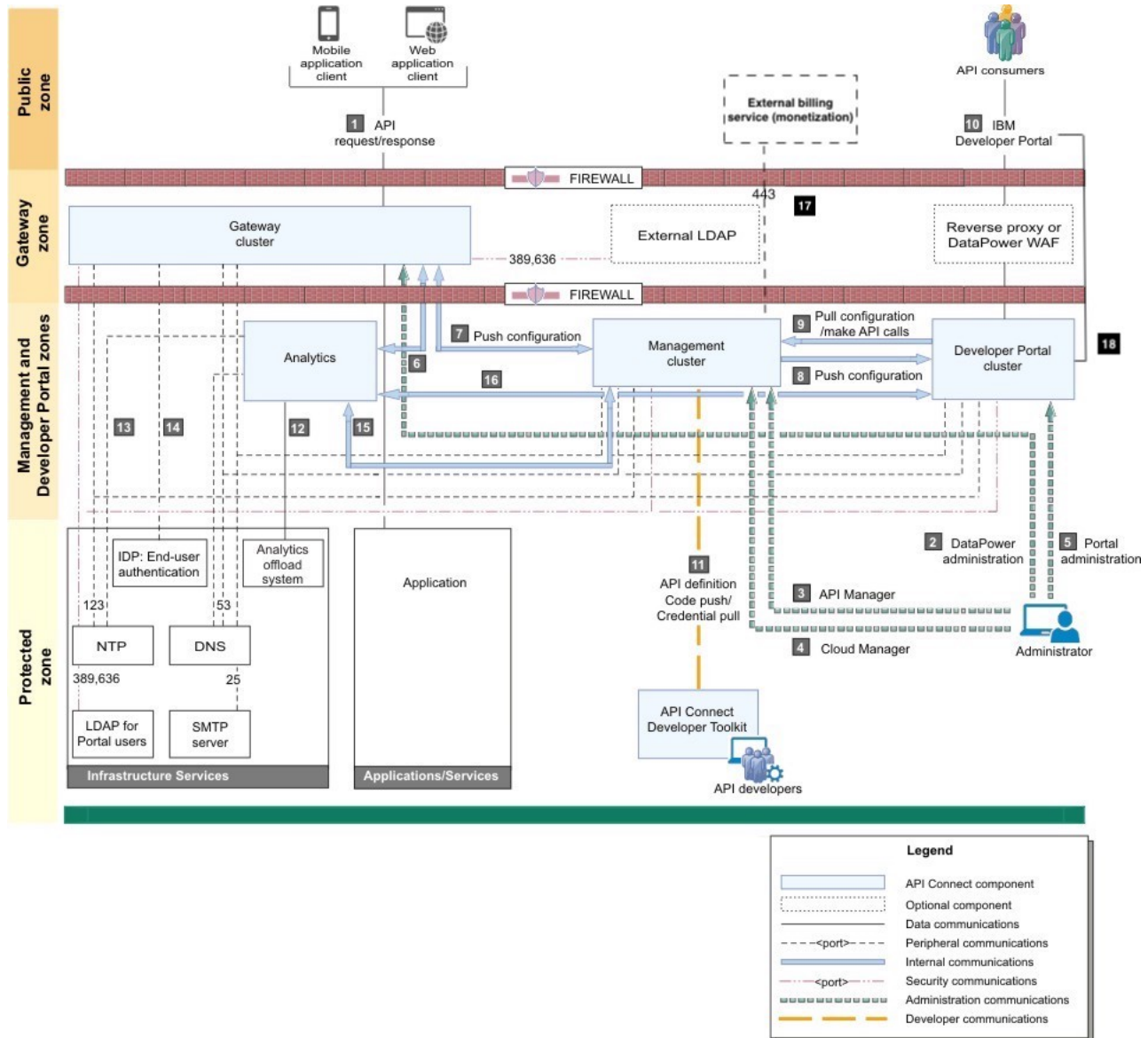


Table 1. Key for the network diagram example. The following table lists the port numbers with a usage description.

	Usage description	Default port number
1	API request/response – Users invoking the provided APIs.	443 HTTPS from Public zone to Gateway zone
2	DataPower® administration – Internal operators who are managing the Gateway servers.	22 SSH, 9090 HTTPS from Protected zone to Gateway zone
3	API Manager – Internal business users who are defining and monitoring APIs.	443 HTTPS from Protected zone to Management zone
4	Cloud Manager – Internal operators who are administering the Cloud.	22 SSH, 443 HTTPS from Protected zone to Management zone
5	Developer Portal administration – Internal operators who are managing the Portal servers.	22 SSH, 443 HTTPS from Protected zone to Management zone
6	Gateway servers post traffic to Analytics service.	443 HTTPS from Gateway servers to Analytics service

	Usage description	Default port number
7	Push configuration – Management servers communicate bi-directionally with Gateway servers.	3000 and 443 HTTPS Management servers to and from Gateway servers for webhook delivery. Port 3000 is the default port for apic-gw-service, and this is the port Management uses to communicate with Gateway. Port 443 is the default port for the platform-api endpoint, which apic-gw-service uses to communicate with Management.
8	Push configuration/webhooks – Management servers push configuration and webhooks to the Developer Portal.	443 HTTPS Management servers to Developer Portal servers for webhook delivery
9	Pull configuration/make API calls – Developer Portal servers pull configuration and call REST APIs.	443 HTTPS from Developer Portal servers to Management servers within Management zone
10	Developer Portal – External developers who are accessing the Developer Portal.	443 HTTPS from Public zone to Developer Portal management zone. The reverse proxy/DataPower WAF for incoming web traffic to the Developer Portal cluster must be a transparent proxy - no modification of the portal URL, port, host name or path is allowed.
11	Push API definition to Management server. Pick up credential for microservice code push.	443 HTTPS from Protected zone to Management zone
12	Analytics offload	Port will depend on type of plugin and protocol used for the offload. Some possible protocols are: HTTP, HTTPS, TCP, UDP, KAFKA
13	Analytics accesses NTP	Standard NTP
14	Analytics access DNS	Standard DNS
15	Management service queries Analytics service	443 HTTPS within Management zone
16	The Portal service invokes an API (GET) on the Analytics service to retrieve data.	443 HTTPS within Management zone
17	External billing service – Management cluster connecting to external billing service (when configured for billing). If you are using Stripe as your external billing service, you must enable connections with the Stripe API at: https://stripe.com/files/ips/ips_api.json . You can also view the Stripe IP addresses at the following URL: https://stripe.com/docs/ips .	443 HTTPS from Management zone to Public zone
18	Developer Portal cluster must be able to access its own site endpoints.	443 HTTPS from Portal zone to Public zone

Firewall port requirements on VMware

The following tables lists ports that must be open in both cluster and non-clustered deployments.

Note that clustered VMware environment deployments require additional ports. See [Firewall enabled ports for clustered OVA deployments](#)

Table 2. Firewall port requirements common to all subsystems

Subsystem	Ports and description
Ports that must be open on all API Connect subsystems	<p>The following ports must be open on the Management Server, Analytics, and Developer Portal subsystems, whether in a cluster or not.</p> <ul style="list-style-type: none"> • 53 (outbound) DNS • 123 (outbound) NTP • 179 (inbound and outbound) BIRD routing daemon - VMware environments (OVA) deployments. Used for communication over TCP between servers either within the same subsystem or across different subsystems. • 443 (inbound and outbound) Used by all subsystems for communication with other subsystems • 2020 (inbound) status request. Used for communication over TCP between servers within the same subsystem. • 9177 (inbound and outbound) Member list (group membership) for API Connect <code>apic</code> daemon communication. Used for communication over both TCP and UDP, between servers within the same subsystem. • 9178 (inbound and outbound) API server, for API Connect <code>apic</code> daemon communication. Used for communication over TCP from between servers, both within the same subsystem and across different subsystems. Also used for communication between the <code>apicup</code> configuration utility and servers (this is the only port used by <code>apicup</code>).

Each subsystem uses ports in addition to the ports in [Table 2](#). See the following table.

Table 3. Additional firewall port requirements for each subsystem

Subsystem	Ports and description
Management Service	<p>Management Service uses the ports in Table 2 plus:</p> <ul style="list-style-type: none"> • 22 remote backup server port (inbound, configurable) If backups are configured to use another port, ensure that the port is open. See Backing up the management subsystem in VMware environments. <p>Note: this port does not have to be open between the management server and the portal server.</p> <ul style="list-style-type: none"> • SMTP server port (outbound), typically 25 • 161 (inbound) SNMP. Over UDP. • LDAP server port (if using LDAP user registry), typically 389 (outbound) • In a clustered deployment, the Management Services uses additional ports. See Firewall enabled ports for clustered OVA deployments. <p>The Automated test behavior application must be able to invoke (outbound) any port that a test system might present an API on. It can be bound to a particular port (if it is the same in all environments) or to a range of ports (to cover all of the ports it might invoke).</p>

Subsystem	Ports and description
Developer Portal	<p>The Developer Portal uses the ports listed in Table 2, plus:</p> <ul style="list-style-type: none"> • 22 - A remote backup server port (inbound, configurable) If backups are configured to use another port, ensure that the port is open. See Backing up and restoring the Developer Portal in a Kubernetes environment. <p>Note: this port does not have to be open between the management server and the portal server.</p> <ul style="list-style-type: none"> • In a clustered deployment, the Developer Portal uses additional ports. See Firewall enabled ports for clustered OVA deployments.
Analytics	<p>The Analytics subsystem uses the ports listed in Table 2, plus:</p> <ul style="list-style-type: none"> • 161 for SNMP is required for both non-clustered and clustered deployments. UDP, inbound only. • In a non-clustered deployment, no additional ports are required. • Analytics supports an optional configuration for offload of data. This configuration might require additional outbound ports to be open. • In a clustered deployment, the Analytics subsystem uses additional ports. See Firewall enabled ports for clustered OVA deployments.
Gateway Server	<p>The Gateway Server uses these ports in both non-clustered and clustered deployments:</p> <ul style="list-style-type: none"> • 161 (inbound) SNMP. UDP protocol. Not needed if SNMP not enabled. • 162 (outbound) SNMP traps • 3000 (inbound) Gateway Service local port (configurable) • 5550 (inbound) XML management port (configurable) • 5554 (inbound) REST management port (if enabled; configurable) • 9022 (inbound) Gateway SSH (if enabled; configurable) • 9090 (inbound) Web GUI console (if enabled; configurable) • 9443 (inbound) Gateway local port (configurable) • In a clustered deployment, the Gateway Server uses additional ports. See Firewall enabled ports for clustered OVA deployments.

Communications inside the Gateway cluster

There are a number of important points to note regarding the communications within the Gateway cluster.

- We advise that you use the same port for all Gateway servers within a cluster.
- Gateway servers communicate with each other to synchronize invocation counts.
- All Gateway servers in a Gateway cluster must be able to reach all of the other Gateway servers in the same Gateway cluster.
- Gateway servers in a Gateway cluster do not directly communicate with Gateway servers in a different Gateway cluster.
- All Gateway servers must be able to reach the management subsystem platform API endpoint, which was configured during the installation of your API Connect environment.

Ethernet interface usage

To separate network traffic, you can use two or more Ethernet interfaces on the DataPower appliance on which a Gateway server is installed. For example, you can use one interface for internal IBM API Connect communications, and another for processing incoming API calls.

Port requirement for Stripe billing service

All Management servers and Developer Portal servers must be able to communicate HTTPS content with the external billing service when billing is configured. If you are using Stripe as your external billing service, you must enable HTTPS communication on port 443 with the Stripe API: https://stripe.com/files/ips/ips_api.json. Note: API Connect does not support using webhooks with Stripe. You can see a list of the Stripe IP addresses at: <https://stripe.com/docs/ips>.

- [Firewall enabled ports for clustered OVA deployments](#)
In a clustered OVA deployment of API Connect, specific ports must be configured for communication between members of each API Connect subsystem.
- [Load balancer ports for clustered OVA deployments](#)
When you deploy a load balancer with a clustered deployment, in a VMware environment, you must ensure that the necessary ports are open.

Related concepts

- [Firewall enabled ports for clustered OVA deployments](#)
- [Load balancer ports for clustered OVA deployments](#)

Firewall enabled ports for clustered OVA deployments

In a clustered OVA deployment of API Connect, specific ports must be configured for communication between members of each API Connect subsystem.

OVA deployments require the common ports that are listed in [Firewall requirements on Kubernetes](#). When the VMs are clustered, additional ports are used for communication between the members of the subsystems in the cluster.

All ports must be enabled inbound and outbound.

Table 1. Firewall enabled ports for clustered VMware environment deployments

Subsystem	Ports
-----------	-------

Subsystem	Ports
Ports that must be open between all subsystem VMs	442, 2379, 2380, 6443, 6444, 9099, 10248, 10249, 10250, 10251, 10252, 10254, 10256, 10257, 10259 These ports must be open between all servers within a given subsystem. For example, from management server to management server, or from portal server to portal server, or from analytics server to analytics server. These ports are not used for communication between subsystems. You might need additional ports for Kubernetes-proxied services. The default range is 30000 – 32767. Since the ports in use can change dynamically, ensure that the default range is open.
Additional ports that must be open between Management Service VMs	Port 8088 is required if the Automated testing behavior application is deployed for the Management Service VMs.
Additional ports that must be open between Developer Portal VMs	3009, 3010, 3306, 3307, 4443, 4444, 4567, 4568, 30865
Additional ports that must be open between Gateway Service VMs	16380, 16381, 26380, 26381
Additional ports that must be open between Analytics VMs	No additional ports are needed.

These internal ports are not used for communication between VMs. Ensure that they are open on the VM server locally.

Table 2. Internal ports reserved by API Connect

Subsystem	Ports
Reserved local ports on all subsystem VMs	8383, 8686, 8080, 8443, 30000:59999
Management Service VMs	2000, 2001, 2002, 2005, 2006, 2007, 2008, 2022, 3003, 3004, 3006, 3007, 3011, 3023, 4150, 4151, 4171, 4222, 5432, 6222, 7777, 8084, 8222, 8404
Portal Service VMs	3058, 3059, 3060, 3061
Analytics VMs	4443, 5000, 5601, 9200, 9300

Note: The ports should support IP-in-IP (protocol 4), per <https://docs.projectcalico.org/getting-started/kubernetes/requirements>.

Load balancer ports for clustered OVA deployments

When you deploy a load balancer with a clustered deployment, in a VMware environment, you must ensure that the necessary ports are open.

The load balancer needs port 443 for all API Connect subsystems (Management, Analytics, Developer Portal, DataPower Gateway).

Load balancer configuration in a VMware deployment

When deploying API Connect for High Availability, it is recommended that you configure a cluster with at least three nodes and a load balancer. A sample configuration is provided for placing a load balancer in front of your API Connect OVA deployment.

About this task

API Connect can be deployed on a single node cluster. In this case the ingress endpoints are host names for which the DNS resolution points to the single IP address of the corresponding node hosting a particular subsystem, and no load balancer is required. For high availability, it is recommended to have at least a three node cluster. With three nodes, the ingress endpoints cannot resolve to a single IP address. A load balancer should be placed in front of an API Connect subsystem to route traffic.

Because it is difficult to add nodes once endpoints are configured, a good practice is to configure a load balancer even for single node deployments. With the load balancer in place, you can easily add nodes when needed. Add the node to the list of servers pointed to by the load balancer to avoid changing the ingress endpoints defined during the installation of API Connect.

To support Mutual TLS communication between the API Connect subsystems, configure the load balancer with **SSL Passthrough** and **Layer 4** load balancing. In order for Mutual TLS to be performed directly by the API Connect subsystems, the load balancer should leave the packets unmodified, as is accomplished by Layer 4. Following is a description of the communication between the endpoints that are configured with Mutual TLS:

- API Manager (with the client certificate portal-client) communicates with the Portal Admin endpoint portal-admin (with the server certificate portal-admin-ingress)
- API Manager (with the client certificate analytics-ingestion-client) communicates with the Analytics Ingestion endpoint analytics-ingestion (with the server certificate analytics-ingestion-ingress)

Note: From v10.0.5.3, it is possible to disable mTLS and use JWT instead, which allows the load-balancers to do TLS termination. For more information, see [Enable JWT instead of mTLS](#).

Set endpoints to resolve to the load balancer

When configuring a load balancer in front of the API Connect subsystems, the ingress endpoints are set to host names that resolve to a load balancer, rather than to the host name of any specific node. For an overview of endpoints, see [Deployment overview for endpoints and certificates](#).

Use this example configuration as a guideline to determine the best way to configure the load balancer for your deployment.

Procedure

- **Appliance deployment**

In this example configuration, the API Connect Management, Portal, Analytics and Gateway subsystems are deployed as three node clusters in Standard mode. An HAProxy load balancer is used. The load balancer is configured with `ssl-passthrough` and with upstream selection based on SNI. DNS resolution is configured to resolve the endpoints to the IP address of the load balancer. If a single HAProxy node is used, then all endpoints must resolve to the IP address of the single HAProxy. The following example endpoints require DNS resolution for this example:

- o `api-manager-ui.sample.example.com`
- o `cloud-admin-ui.sample.example.com`
- o `consumer-api.sample.example.com`
- o `platform-api.sample.example.com`
- o `admin.portal.sample.example.com`
- o `web.portal.sample.example.com`
- o `analytics.ingestion.sample.example.com`
- o `api-gateway.sample.example.com`
- o `apic-gw-service.sample.example.com`

Following is an example HAProxy configuration for one HAProxy node distributing traffic to Management and Portal clusters:

Note:

- o When you configure a load balancer in front of a Management subsystem, specify timeouts of at least 240 seconds. Note that large deployments might need larger values.

The default timeout is typically 50 or 60 seconds, which is not long enough to avoid **409**

Conflict or **504 Gateway Timeout** errors. The **409 Conflict** error can occur when the time needed to complete an operation is sufficiently long that a second request gets issued.

For example, to specify 240 seconds when using HAProxy as a load balancer, set `timeout client` and `timeout server` to `240000`.

- o Ensure that the load balancer is configured to support the cipher `TLS_ECDHE_RSA_WITH_AES256_GCM_SHA384 (0xc030)`. When the Developer Portal, as a client, calls back to `platform-api-endpoint` or `consumer-api-endpoint` of the management subsystem, it sends only a single cipher in the Client Hello. The cipher is `TLS_ECDHE_RSA_WITH_AES256_GCM_SHA384 (0xc030)`. The SSL Handshake will fail if this cipher is not configured (supported) on the client's load balancer.

```
# This sample HAProxy configuration file configures one HAProxy node to distribute traffic to
# Management, Portal, Analytics, and Gateway clusters. Another option is to configure one HAProxy
# node per cluster.
```

```
global
```

```
log /dev/log      local0
log /dev/log      local1 notice
chroot /var/lib/haproxy
stats socket /run/haproxy/admin.sock mode 660 level admin
stats timeout 30s
user haproxy
group haproxy
daemon
```

```
# Default SSL material locations
ca-base /etc/ssl/certs
crt-base /etc/ssl/private
```

```
# Default ciphers to use on SSL-enabled listening sockets.
# For more information, see ciphers(1SSL). This list is from:
# https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:ECDH+3DES:DH+3DES:
RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!MD5:!DSS
ssl-default-bind-options no-ssl3
```

```
defaults
```

```
log          global
mode        http
option      httplog
option      dontlognull
timeout connect 5000
timeout client 240000
timeout server 240000
errorfile 400 /etc/haproxy/errors/400.http
errorfile 403 /etc/haproxy/errors/403.http
errorfile 408 /etc/haproxy/errors/408.http
errorfile 500 /etc/haproxy/errors/500.http
errorfile 502 /etc/haproxy/errors/502.http
errorfile 503 /etc/haproxy/errors/503.http
errorfile 504 /etc/haproxy/errors/504.http
```

```
##### MANAGEMENT CONFIGURATION #####
```

```
frontend fe_management
```

```
mode tcp
option tcplog
#
# Map to the hostname and TCP port for the Management load balancer.
# In this example, the hostname for the load balancer is ubuntu.sample.example.com.
#
bind ubuntu.sample.example.com:443
tcp-request inspect-delay 5s
tcp-request content accept if { req_ssl_hello_type 1 }
```

```
#
# The value for the Management endpoints as defined in the apiconnect-up.yml
# file using the apicup installer. In this example, the endpoints are api-manager-ui.sample.example.com,
# cloud-admin-ui.sample.example.com, consumer-api.sample.example.com, and
# platform-api.sample.example.com. Standard SNI structure specifies
# whether the INCOMING request is for api-manager or cloud-admin or for consumer-api or platform-api
# then use "be_management".
```

```

#
use_backend be_management if
{ req_ssl_sni -i api-manager-ui.sample.example.com OR req_ssl_sni -i cloud-admin-ui.sample.example.com }
use_backend be_management if
{ req_ssl_sni -i consumer-api.sample.example.com OR req_ssl_sni -i platform-api.sample.example.com }

#
# be_management is defined to point management traffic to the cluster
# containing three management nodes
#

backend be_management
mode tcp
option tcplog
balance roundrobin
option ssl-hello-chk

#
# One entry per Management node in the cluster.
# Hostname and TCP Port for each Management node.
#
server management0 manager1.sample.example.com:443 check
server management1 manager2.sample.example.com:443 check
server management2 manager3.sample.example.com:443 check

##### PORTAL CONFIGURATION #####
frontend fe_portal
mode tcp
option tcplog
#
# The Hostname and TCP Port for the Portal Load balancer
#
bind ubuntu.sample.example.com:443
tcp-request inspect-delay 5s
tcp-request content accept if { req_ssl_hello_type 1 }

#
# The value for both of the Portal subsystem endpoints as defined in the apiconnect-up.yml file
#
use_backend be_portal if
{ req_ssl_sni -i admin.portal.sample.example.com OR req_ssl_sni -i web.portal.sample.example.com }

backend be_portal
mode tcp
option tcplog
balance roundrobin
option ssl-hello-chk

#
# One entry per Portal node.
# Hostname and TCP Port for the Portal node.
#
server portal0 portal1.sample.example.com:443 check
server portal1 portal2.sample.example.com:443 check
server portal2 portal3.sample.example.com:443 check

##### ANALYTICS CONFIGURATION #####
frontend fe_analytics
mode tcp
option tcplog
#
# The Hostname and TCP Port for the Analytics Load balancer
#
bind ubuntu.sample.example.com:443
tcp-request inspect-delay 5s
tcp-request content accept if { req_ssl_hello_type 1 }

#
# The value for both of the Analytics subsystem endpoints as defined in the apiconnect-up.yml file
#
use_backend be_analytics if
{ req_ssl_sni -i analytics.ingestion.sample.example.com }

backend be_analytics
mode tcp
option tcplog
balance roundrobin
option ssl-hello-chk

#
# One entry per Analytics node.
# Hostname and TCP Port for the Analytics node.
#
server analytics0 analytics1.sample.example.com:443 check
server analytics1 analytics2.sample.example.com:443 check
server analytics2 analytics3.sample.example.com:443 check

##### GATEWAY CONFIGURATION #####
frontend fe_gateway
mode tcp
option tcplog
#
# The Hostname and TCP Port for the Gateway Load balancer

```

```

#
bind ubuntu.sample.example.com:443
tcp-request inspect-delay 5s
tcp-request content accept if { req_ssl_hello_type 1 }

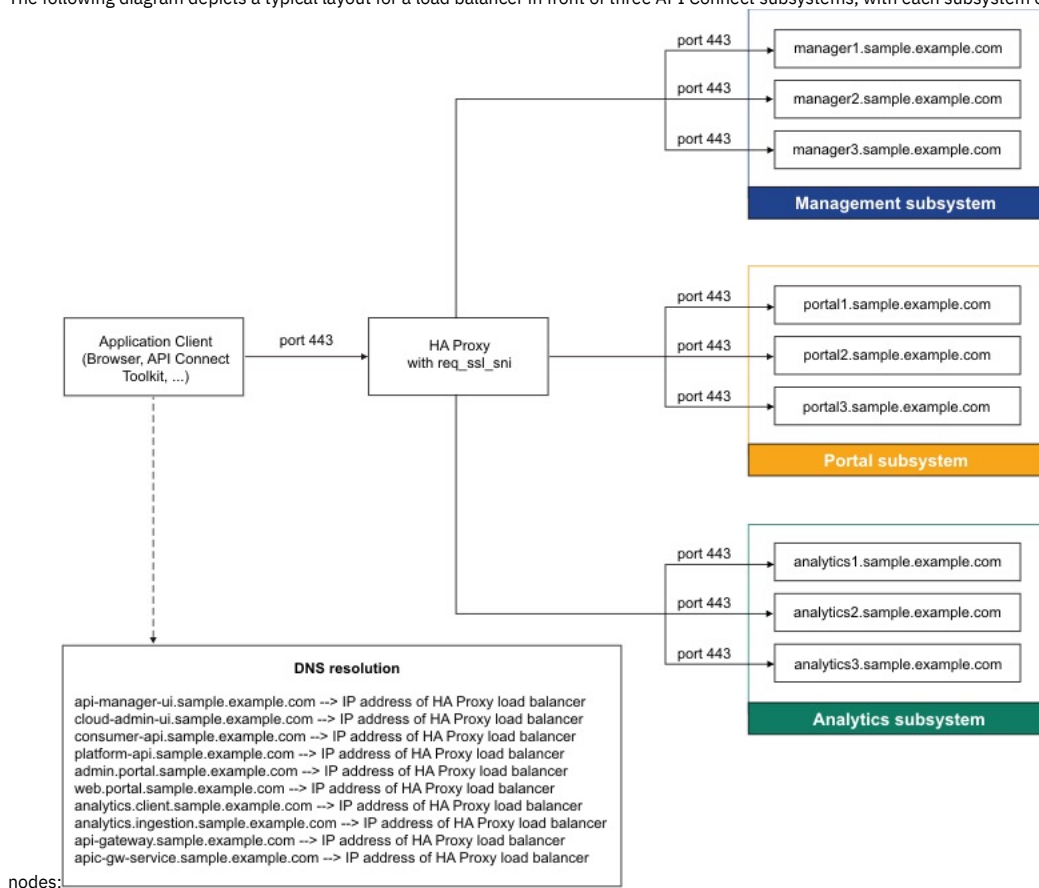
#
# The values for the Gateway subsystem endpoints as defined in the apiconnect-up.yml file.
#
use_backend be_gateway if
{ req_ssl_sni -i api-gateway.sample.example.com OR req_ssl_sni -i apic-gw-service.sample.example.com }

backend be_gateway
mode tcp
option tcplog
balance roundrobin
option ssl-hello-chk

#
# One entry per Gateway node.
# Hostname and TCP Port for the Gateway node.
#
server gateway0 gateway1.sample.example.com:443 check
server gateway1 gateway2.sample.example.com:443 check
server gateway2 gateway3.sample.example.com:443 check

```

- The following diagram depicts a typical layout for a load balancer in front of three API Connect subsystems, with each subsystem containing three Appliance/OVA



A two data center deployment strategy on VMware

An overview of the two data center disaster recovery deployment strategy in API Connect.

Key points of the two data center disaster recovery (DR) solution

- Two data center DR is an active/warm-standby deployment for the API Manager and Developer Portal services, and must use manual failover.
- Two DataPower® Gateway subsystems must be deployed to provide high availability for the gateway service. However, this scenario doesn't provide high availability for the analytics service.
- If high availability is required for the analytics service, two analytics subsystems must be configured, one per gateway subsystem, but with this configuration Developer Portal analytics isn't possible.
- Data consistency is prioritized over data availability.
- Latency between the two data centers must be less than 80 ms.
- Replication of the API Manager is asynchronous, so it is possible that the most recent updates do not transfer to the warm-standby data center if there is an active data center failure.
- Replication of the Developer Portal is synchronous, and therefore the latency is limited to 80 ms or less.

- The API Manager and the Developer Portal services in the two data centers must use the same deployment profile.
- The deployment in each data center is effectively an instance of the same API Connect deployment - therefore, the endpoints and certificates must all be the same.
- It is not possible to use the Automated API behavior testing application ([Installing the Automated API behavior testing application](#)) in a two data center disaster recovery configuration.

Deployment architecture

A two data center deployment model is optimized for data consistency ahead of data availability, and must use manual failover when a fault occurs. For high availability of Management and Portal subsystems ensure that you use a three replica deployment profile. For more information on deployment profiles, see: [Planning your deployment topology and profiles](#).

To achieve high availability for the DataPower Gateway, you must deploy two gateway subsystems. One subsystem in the active data center, and a separate subsystem in the warm-standby data center. Publish all Products and APIs to both gateway subsystems. The gateway subsystems are independent, and so are insulated if an issue occurs in one of them. A global dynamic router can then be used to route traffic to one gateway subsystem or the other. If high availability is also required for the analytics service, two analytics subsystems must be configured, one per gateway subsystem, but with this configuration Developer Portal analytics isn't possible.

Dynamic routing

Note: A dynamic routing device, such as a load balancer, is required to route traffic to either data center. However, neither this device nor its configuration is part of the API Connect offering. Contact IBM Services if you require assistance with configuring a dynamic router.

The dynamic router configuration must handle the traffic between the subsystems and between the data centers. For example, the consumer API calls from the Developer Portal to API Manager must transfer through a dynamic router so that the Developer Portal can use a single endpoint regardless of which data center API Manager is active in. The same is needed for calls to the Platform and Admin APIs from the other subsystems, as well as for incoming UI traffic for the Developer Portal UI, Cloud Manager UI, and API Manager UI.

The dynamic router must support SSL passthrough, so that it routes the Mutual TLS (mTLS) connections between API Manager and the Developer Portal, and between API Manager and the DataPower Gateway. The router should not do TLS termination, it should do layer 4 based routing by using SNI.

Note: From v10.0.5.3, it is possible to disable mTLS and use JWT instead, which allows the load-balancers to do TLS termination. For more information, see [Enable JWT instead of mTLS](#).

Service status

When a failure occurs, it is common practice to display an interstitial system outage web page. The dynamic router can be configured to display this web page when there is a failure, while the manual failover to the warm-standby data center is taking place.

Deployment profiles

Both one replica and three replica deployment profiles can be used with two data center DR, but they must be the same at each data center. For more information about the deployment profiles, see [Requirements for initial deployment on VMware](#).

For more information about a two data center DR deployment, see the following topics:

VMware

- [Installing a two data center deployment on VMware](#)
- [Maintaining a two data center deployment](#) (including information about normal operation data flows)
 - [Failure handling of a two data center deployment](#)
 - [Recovering from a failover of a two data center deployment](#)
 - [Backup and restore considerations for a two data center deployment](#)
 - [Upgrading a two data center deployment](#)

Planning your Analytics deployment

Plan your IBM® API Connect Analytics deployment by reviewing options for topology, data modifications, and storage.

Planning your Analytics topology is an important step in ensuring a successful and scalable solution. After you plan your overall API Connect topology, you should plan the Analytics subsystem deployment by choosing a topology, planning any data modifications, and estimating the amount internal storage that is needed for the deployment.

- [Planning the Analytics deployment profile](#)
Begin planning your API Connect Analytics deployment by choosing a deployment profile.
- [Planning to offload data to a third-party system](#)
Review the considerations for offloading data from your IBM API Connect Analytics deployment to a third-party system for additional storage and analysis.
- [Planning to disable internal storage](#)
Review considerations for disabling local storage and offloading all data in your API Connect Analytics deployment.
- [Planning to modify your analytics data](#)
Review how you can optionally customize the analytics data in your API Connect Analytics deployment before storing it.
- [Estimating internal storage space](#)
If you plan to store data locally in your IBM API Connect Analytics deployment, estimate disk space requirements.

Planning the Analytics deployment profile

Begin planning your API Connect Analytics deployment by choosing a deployment profile.

Deployment profile

The first decision you must make is which **deployment-profile** you want to use. The available deployment profiles for analytics are: **n1xc2.m16, n1xc4.m32, n1xc6.m48, n3xc4.m16, n3xc6.m48, n3xc8.m64**.

For more information on deployment profiles, see [Planning your deployment topology and profiles](#).

Planning to offload data to a third-party system

Review the considerations for offloading data from your IBM® API Connect Analytics deployment to a third-party system for additional storage and analysis.

The analytics event data that is generated by the Gateway and collected by the analytics subsystem can be offloaded to a third-party service in real-time. The third-party service can provide enhanced monitoring, alerting, storage, and/or analysis, in addition to what the built-in Analytics subsystem provides. Multiple target systems are supported, and the data can be modified prior to offloading.

APIC supports offloading analytics data to the following third-party systems:

- HTTP(s) endpoints
- OpenSearch
- Elasticsearch
- Apache Kafka
- Syslog servers

For instructions on configuring your offload settings, see [Offloading data to a third-party system](#).

If you do not need internal storage and only want to use the third-party system for working with your analytics data, consider disabling internal storage entirely as explained in [Planning to disable internal storage](#).

Planning to disable internal storage

Review considerations for disabling local storage and offloading all data in your API Connect Analytics deployment.

The Analytics subsystem provides a complete solution for routing, storing, and viewing the analytics data. However, you might not need the internal storage and viewing solution, especially if you are primarily using a third-party system for data offloading. In this scenario, you can greatly reduce your CPU and memory costs by disabling internal storage.

Important: It is not possible to change this configuration after installation. You must decide on whether you want to disable internal storage before installation. With this approach, data that is offloaded is not viewable in the API Manager and Developer Portal UIs.

For more information on disabling internal storage, see [Disabling internal storage](#).

Planning to modify your analytics data

Review how you can optionally customize the analytics data in your API Connect Analytics deployment before storing it.

You can create filters to customize the analytics data before storing in external or internal storage. Filters allow you to add new fields, modify existing fields, or remove fields from the data. You can do this globally for the entire Analytics subsystem, or you can do it conditionally for a specific API. By default, filters are defined at the global level, so unless you include conditionals, changes are applied to every piece of data flowing through the pipeline.

For information on adding filters to the Analytics configuration to modify your data, see [Modifying your analytics data](#).

Note: If you want to modify the data for specific APIs rather than all incoming data, you can use gateway scripts or the `set-variable` policy, as described here: [Modifying analytics API event data](#).

What fields can you modify?

The fields that are available for you to interact with are described in [API event record fields](#). Each API defined in the Management subsystem and published to your Gateway has its own success and error log policies. The log policy settings determine what data is available for customization. Be sure to verify the existence of a field before trying to modify it; otherwise your pipeline might fail.

Modifying fields is a complex operation and can cause problems with your data. It is highly recommended that you remove fields entirely (instead of modifying them) if you need to sanitize data. If you want to change the format of an existing field, you should create a new, unique field.

Important: Do not change the data type of existing fields or modify fields critical to the Analytics subsystem's operations and access control. Doing so can cause problems with retrieving or storing data. The following fields are restricted:

- org_id
- catalog_id
- space_id
- developer_org_id
- datetime
- @timestamp

The service for the data pipeline is based on Elastic Logstash, which provides [a set of predefined filter plugins](#) that you can use for modifying data. However, the plugins are third-party software that IBM does not control, so IBM cannot guarantee support for them.

Estimating internal storage space

If you plan to store data locally in your IBM® API Connect Analytics deployment, estimate disk space requirements.

This information applies only to data that is stored in API Connect Analytics and not to data that is offloaded to a third-party system. The formula and guidelines are based on known information about how the Analytics service stores data, and cannot be directly applied to any other storage system.

Use the following guidelines to calculate a rough estimate of the amount disk space you need for storing stateful data in the API Connect Analytics subsystem.

Data storage: calculate how much data you want to store

The amount of disk space needed for storing analytics data is determined by the following factors:

Number of copies of the analytics data

Each copy of the analytics data must live on a separate node. The number of nodes in your deployment (based on your chosen profile in [Planning the Analytics deployment profile](#)) determines the number of copies of analytics data that is stored in your deployment. With the three replica deployment profile, analytics data is replicated to three nodes (two replicas and one primary copy of the data) for storage. In the one replica profile, you only need to store one copy of the data.

Number of copies of data:

```
Copies of data = [1 | 3]
```

Number of days that data is retained

By default, analytics data is retained for 90 days, but you can [modify the retention setting](#) as needed. Make sure you know how long you want to store the data before attempting to calculate required disk space.

Number of days that data is retained:

```
Data retention = [90 days | preferred length]
```

Amount of each type of data stored

The required storage space for each type of logging is highly dependent on your APIs and usage. For each API, you can [Configure activity logging](#) for the API activity, header, and payload. You can also [customize the data](#) to add, redact, or remove fields, which also impacts the amount of data that you store.

To estimate storage needs, calculate the average size of each type of log. When calculating your estimates, remember that the header logging size is the sum of activity logging size and the average size of your headers. The payload logging size is the sum of the header logging size and the average size of your payloads. Typically the average size of an activity logged event is 600-1000 bytes depending on the uniqueness and complexity of your analytics data. This number is highly dependent on your APIs and your implementations. For a rough estimate, you can use an average of 800 bytes per activity logged event.

If you choose to add fields, calculate the average size of the new fields as well, and add that number to all types of log policies. If you choose to remove fields, you should not subtract this size from the log policies unless you are also removing headers and/or payloads.

Amount of each type of data that is stored:

```
Activity log bytes per call = [600-1000 bytes]
Header log bytes per call = Activity log bytes per call + Average size of headers
Payload log bytes per call = Header log bytes per call + Average size of payloads
```

Percentage of each type of data

Estimate the percentage of each type of log (activity, header, payload) for all API calls.

If you follow best practices of using only activity logging for production environments and using only payload logging for test environments, this number is easy to determine. If you use different log policies per API, and they depend on "success" or "error" factors, the percentage is more difficult to determine. Typically, if you do not use an all-or-nothing method for logging, error rates range from 3% to 25% with subsequent payload logging in test and production environments. However, this is entirely dependent on your use case and your APIs.

Percentage of each type of data that is stored:

```
% of Activity log = [0, 100 or other estimate]
% of Header log = [0, 100 or other estimate]
% of Payload log = [0, 100 or other estimate]
```

Estimated number of API calls per month

When planning your API Connect deployment, this number is helpful. When estimating analytics storage, this number is vital because it is directly correlated to how much storage you need for your deployment.

If you do not know the number of calls per month, but you do know the number of calls per second, use the following formula to convert it to calls per month:

```
Calls per month = Calls per second * 86400 seconds per day * 30 days per month
```

Number of API calls per month:

```
Calls per month = [any estimate]
```

Calculating your disk space requirement

Estimate the disk space requirement for each storage node by completing following calculations.

Formula

```
Bytes per call = (% of Activity Log * Activity Log bytes per call) + (% of Header Log * Header Log bytes per call) + (% of
Payload Log * Payload Log bytes per call)
Calls per retention period = Calls per month * (Number of days retained / 30 days per month)
Storage for API calls = Calls per retention period * Bytes per call * Copies of data
Total storage = Storage for API calls + Overhead
Storage per node = Total storage / Nodes
```

Details

1. Bytes per call:

Estimate the number of bytes that are logged for a single copy of each API call. This can be calculated from the prerequisites of the percentage of each data type and the amount of each data type stored.

```
Bytes per call = (% of Activity Log * Activity Log bytes per call) + (% of Header Log * Header Log bytes per call) +
(% of Payload Log * Payload Log bytes per call)
```

2. Calls per retention period:

Calculate the anticipated number of API calls per retention period. This can be calculated from the prerequisites of the estimated calls per month and your desired data retention.

```
Calls per retention period = Calls per month * (Number of days retained / 30 days per month)
```

3. Storage for API Calls:

Calculate the storage needed for all copies of your api calls for your retention period. This can be calculated from steps 1 and 2, as well as the prerequisite of your total copies of your data.

```
Storage for API calls = Calls per retention period * Bytes per call * Copies of data
```

4. Total Storage:

Calculate the total storage you need by adding buffer space to the value from step 3. The Analytics service requires some overhead space to complete its operations; for example, to contain system data and temporary debug header or payload logging. In addition, allowing extra space provides a buffer in case you underestimated the number of calls, or experience an unexpected increase.

There is no specific value for the buffer because it's based on your own situation. One approach is to use a value that rounds the Storage for API calls result from step 3 to the next whole number. Make sure the rounding leaves you with a comfortable amount of additional space. For example, if the result from step 3 is 380 GB, then adding 20 GB to reach 400 GB is probably not sufficient and you should consider rounding to a larger value such as 500 GB.

```
Total storage = Storage for API calls + Buffer
```

5. Storage per node:

Calculate the total storage amount required per storage node. The number of storage nodes is dependent on your deployment profile. For the one replica profile, use 1. For the three replica profile, it defaults to 3.

```
Storage per node = Total storage / Nodes
```

Remember: This result is only an estimate. You should monitor the use of space over time and adjust storage as needed.

Save this information for when you are completing the installation steps. You specify the disk size when you deploy the OVA file: [Deploying the Analytics subsystem OVA file](#).

Example

Deployment information:

```
Copies of data = 3
Data retention = 90 days
Activity log bytes per call = 850 bytes
Header log bytes per call = 15k bytes
Payload log bytes per call = 30.5k bytes
% of Activity log = 100%
% of Header log = 0%
% of Payload log = 0%
Calls per month = 64 million
```

Formula:

```
Bytes per call = (% of Activity Log * Activity Log bytes per call) + (% of Header Log * Header Log bytes per call) + (% of
Payload Log * Payload Log bytes per call)
Calls per retention period = Calls per month * (Number of days retained / 30 days per month)
Storage for API calls = Calls per retention period * Bytes per call * Copies of data
Total storage = Storage for API calls + Overhead
Storage per node = Total storage / Nodes
```

Details:

1. Bytes per call = 850 bytes

```
(100% of Activity Log * 850 bytes) + (0% of Header Log * 15k bytes) + (0% of Payload Log * 30.5 bytes)
```

2. Calls per retention period = 192 million calls

```
64 million calls per month * (90 days retained / 30 days per month)
```

3. Storage for API calls = 489.6 GB

```
192 million calls per period * 850 bytes per call * 3 copies of data
```

4. Total storage = 600 GB

```
489.6GB storage + Buffer
```

Since rounding to 500 GB only provides 10.4 GB of extra space, consider rounding to 600 GB instead.

5. Storage per node = 200 GB

```
600GB Total storage / 3 nodes
```

In this example, the estimated disk needed on each node is 200GB.

Working with certificates

Use the `certs` command included in the APICUP installer to set and manage certificates for each subsystem. Default certificates are automatically applied, but defaults can be overridden by user-supplied custom certificates.

About this task

- [Certificate management: Read This First](#)
Requirements and best practices for managing API Connect certificates.
- [Setting and managing certificates](#)
Default certificates are automatically applied, but defaults can be overridden by custom certificates.
- [Reference for certificates, commands, and validations](#)
This section contains the reference information for certificates, commands for working with certificates, and validations.

Certificate management: Read This First

Requirements and best practices for managing API Connect certificates.

Important: Customization of public certificates and public user-facing certificates is recommended. Customization of internal certificates is strongly discouraged. For management purposes, API Connect certificates can be grouped by type (default, custom, and common) and by usage (public, public and user-facing, and internal). Understand each type and usage before you change any certificates.

- A default certificate is a private certificate that is uniquely generated by the installer for the current project directory, and will pass validation. Default certificates are automatically generated for each subsystem by the `apicup subsys install` command, unless the certificates were explicitly set by using the `apicup certs set` command. Note that the default certificates are self-signed, so they might not provide a level of trust suitable for external communication.
- For optimal trust levels, we recommend that you explicitly set all public and user-facing certificates by creating custom certificates.
- Some certificates are common across subsystems. Subsystems require the common certificates to allow them to register with the management subsystem. When installing any one subsystem, the common certificates are set for that subsystem and for all the other subsystems. If you use custom certificates for the common certificates, the custom certificates must be set prior to setting any other custom certificates.
- We do not recommend the explicit setting of internal certificates. The changing of internal certificates creates a risk of incompatibility with other internal certificates.

To review the usage (public, public and user-facing, or internal) of each certificate, see the following [Certificate management best practice](#) table.

Table 1. Certificate management best practice

Certificate usage	Certificate name	Best practice management
Public and User-facing	<ul style="list-style-type: none"> • platform-api, api-manager-ui, cloud-admin-ui • consumer-api • portal-www-ingress • hub-endpoint • turnstile-endpoint • management-replication-ingress • portal-replication-ingress 	For optimal trust levels, set these explicitly. Recommended to be explicitly set as custom certificates because they are presented to an end user through a browser or Command Line Interface (CLI).
Internal	<ul style="list-style-type: none"> • root-ca, ingress-ca • portal-admin-ingress, portal-client • analytics-ingestion-ingress, analytics-ingestion-client <p>For VMware appliance deployments only: k8s-ca, appliance-client</p>	Do not change. Accept the default certificates. It is possible to change these certificates (except ingress-ca) but is strongly discouraged because you risk creating incompatibilities that can block internal communications. Each intermediate cert is used to generate other internal certs. If, for example, you change an intermediate cert, the certs generated from it might not work with internal certs generated from other intermediate certs. Note that ingress-ca is auto-generated and cannot be set using the <code>apicup certs set</code> command.

See also:

- [Setting and managing certificates](#)
- [Reference for certificates, commands, and validations.](#)

Setting and managing certificates

Default certificates are automatically applied, but defaults can be overridden by custom certificates.

About this task

- [Setting default certificates](#)
Default certificates are automatically generated by APICUP when the subsystem is installed.
- [Setting custom certificates](#)
Use the APICUP installer `certs` commands to set custom certificates.
- [Replacing custom certificates](#)
Use the APICUP installer `certs` commands to replace existing certificates.
- [Setting common certificates](#)
Common certificates are set for one subsystem, but are applied to all subsystems. Use the APICUP installer `certs` commands to set the common certificates.
- [Setting the encryption-secret for the management database](#)
Use the APICUP installer `certs` commands to set the encryption-secret for the management database.
- [Clearing certificates](#)
Certificates can be cleared in order to set new certificates.

Setting default certificates

Default certificates are automatically generated by APICUP when the subsystem is installed.

About this task

Important:

- Customization of public certificates and public user-facing certificates is recommended. Customization of internal certificates is strongly discouraged.
- To view a list of public, public user-facing, and internal certificates, see [Certificate management: Read This First](#). For details on each certificate, see [Certificate reference](#).

Default certificates are generated for each subsystem by the `apicup subsys install` command. If certificates are not explicitly set by using the `apicup certs set` command, then default certificates are automatically generated by APICUP. The default certificates are self-signed, so they might not provide a level of trust suitable for external communication.

Procedure

1. Enter the settings for the subsystem by using `apicup subsys set <SUBSYS>` and validate the subsystem settings by using `apicup subsys get <SUBSYS> --validate`. The subsystem must pass validation before setting the certificates.
2. Install the subsystem by using the `apicup subsys install` command.
3. The default certificates are created for the subsystem. A default certificate is a private certificate that is uniquely generated by the installer for this project directory, they are self-signed and always pass validation.
4. List all certificates that are set for a subsystem by using the `apicup certs list` command.

```
apicup certs list -help
```

List all configured certificates

Usage:

```
apicup certs list SUBSYS [flags]
```

Flags:

```
-h, --help help for list
```

Global Flags:

```
--accept-license Accept the license for API Connect
--debug Enable debug logging
```

Following is example output from the `apicup certs list <SUBSYS>` command:

```
Common certificates
=====
```

Name	Summary	Validation errors
analytics-ingestion-client	CN: analytics-ingestion-client SubjectKeyId: 27:60:BF:DF:6C:34:29:FE:8E:83:21:1B:C0:14:B2:9E AuthorityKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E	
ingress-ca	CN: ingress-ca SubjectKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E AuthorityKeyId: 5E:6D:5C:6E:2C:BE:50:F3:4E:EE:FD:02:76:86:6C:5A	
portal-client	CN: portal-client SubjectKeyId: 08:A8:57:A5:99:BC:79:FA:14:59:A4:98:6D:F7:43:C4 AuthorityKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E	
root-ca	CN: root-ca SubjectKeyId: 5E:6D:5C:6E:2C:BE:50:F3:4E:EE:FD:02:76:86:6C:5A AuthorityKeyId:	

```
Subsystem mgmt certificates
=====
```

Name	Summary	Validation errors
api-manager-ui	CN: api-manager-ui SubjectKeyId: A3:C1:A1:4F:21:23:21:2F:1F:D7:87:30:E1:1E:33:A3 AuthorityKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E	
appliance-client	CN: appliance-client SubjectKeyId: 5C:FB:0F:5D:B8:BF:6F:89:CB:25:DD:54:31:A7:B4:63 AuthorityKeyId: 60:D9:B2:37:0B:17:FB:CD:FC:49:29:32:F6:A6:49:7C	
cloud-admin-ui	CN: cloud-admin-ui SubjectKeyId: E7:E2:D6:35:95:6B:D4:3B:F7:F7:9F:5F:DD:B8:02:E9 AuthorityKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E	
consumer-api	CN: consumer-api SubjectKeyId: 2A:80:EB:A6:31:9E:A5:C6:41:D9:1F:69:D1:9E:31:75 AuthorityKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E	
encryption-secret k8s-ca	2D:F9:61:0C:45:CB:6E:90:85:E0:0E:D3:DF:CC:B4:47 CN: k8s-ca SubjectKeyId: 60:D9:B2:37:0B:17:FB:CD:FC:49:29:32:F6:A6:49:7C AuthorityKeyId:	
platform-api	CN: platform-api SubjectKeyId: 6D:E7:60:21:81:7E:F6:40:A4:9A:2F:88:35:D1:18:04	

Setting custom certificates

Use the APICUP installer `certs` commands to set custom certificates.

About this task

Important:

- Customization of public certificates and public user-facing certificates is recommended. Customization of internal certificates is strongly discouraged.
- To view a list of public, public user-facing, and internal certificates, see [Certificate management: Read This First](#). For details on each certificate, see [Certificate reference](#).

The APICUP installer can be used to set certificates for each subsystem during installation. If certificates are not explicitly set by using the `apicup certs set` command, then default certificates are generated by APICUP. The default certificates are self-signed, so they might not provide a level of trust suitable for external communication.

Requirements for custom certificates:

- Extended Key Usage (EKU), either `serverAuth` or `clientAuth` depending upon the type of certificate. Certificates of type *Server* must have an Extended Key Usage with `serverAuth` purpose. Certificates of type *Client* must have an Extended Key Usage with `clientAuth` purpose.
- Subject Alternative Name (SAN) for the required hosts
- Any custom common certificates that are being used must be set prior to setting any custom certificates for a subsystem.

See [Certificate reference](#) to view the list of common certificates and to determine whether an EKU is needed for a certificate and which type of EKU (`serverAuth` or `clientAuth`).

Certificates and identical endpoints:

The Management subsystem has four public endpoints: `api-manager-ui`, `cloud-admin-ui`, `platform-api`, and `consumer-api`. Distinct TLS certificates can be set for each endpoint. However, if any two endpoints are identical, only one TLS certificate will be effective. When 2 or more endpoints are set to the same host, the secrets associated with the endpoints should be the same or contain the same certificate files.

Note: Once API Connect has been installed (meaning that the `apicup subsys install SUBSYS` command has been executed) with a given set of certificates, only the certificates for the public ingress endpoints (`portal-www`, `api-manager-ui`, `cloud-admin-ui`, `platform-api`, `consumer-api`) can be modified. The TLS certificates involved in mutual authentication (`portal-admin-ingress`, `portal-client`, `analytics-ingestion-ingress`, `analytics-ingestion-client`) cannot be modified after the `install` command has been executed.

Procedure

1. Set up and validate the subsystem. Enter the settings for the subsystem using `apicup subsys set <SUBSYS>` and validate the subsystem settings using `apicup subsys get <SUBSYS> --validate`. The subsystem must pass validation before setting the certificates.
2. Generate the custom certificate with the appropriate EKU and SAN. You will need to obtain the private key, public certificate, and CA certificates in non-password-protected PEM format for the custom certificate. Following is an example for how to generate a certificate (`platform-api-example`) with an EKU `serverAuth` and SAN using `openssl`:

```
openssl x509 -req -days 360 -in platform-api-example.csr -CA root-ca.pem -CAkey root-ca-key.pem -CAcreateserial -out platform-api-cert -sha256 -extfile <(cat /etc/ssl/openssl.cnf <(printf "\n[SAN]\nsubjectAltName=DNS:fqdn.myserver.com\nextendedKeyUsage=serverAuth") -extensions SAN
```

where

- `DNS:fqdn.myserver.com` is the fully qualified domain name of the endpoint the certificate applies to. This matches the endpoints entered in the APICUP installer. See [Deploying the Management virtual appliance](#)
- `platform-api-example.csr` is the file name for the certificate signing request

Following is an example for how to generate a certificate (`portal-client`) with an EKU `clientAuth` and SAN using `openssl`:

```
openssl x509 -req -days 360 -in portal-client-example.csr -CA root-ca.pem -CAkey root-ca-key.pem -CAcreateserial -out portal-client-cert -sha256 -extfile <(cat /etc/ssl/openssl.cnf <(printf "\n[SAN]\nkeyUsage=critical,digitalSignature,keyEncipherment\nextendedKeyUsage = clientAuth\nbasicConstraints=critical,CA:FALSE\nsubjectKeyIdentifier=hash\n")) -extensions SAN
```

3. Once the certificate has been created, set the certificate by entering the following command:
`apicup certs set SUBSYS CERT_NAME [CERT_FILE KEY_FILE CA_FILE] [KEY_FILE] [flags]`

If the certificate is signed by an intermediate CA, the `CA_File` argument must point to a file that concatenates the intermediate CA, followed by the root CA, in that order.

You can find definitions for commands at the following location: [Command reference](#)

If the certificate was generated with an EKU `serverAuth`, it must be assigned to a server certificate. If the certificate was generated with an EKU `clientAuth`, it must be assigned to a client certificate.

4. Repeat for additional custom certificates.
5. After setting the custom certificates, you can optionally generate the remaining default certificates prior to installation by entering the `apicup certs generate` command. The `generate` command generates any certificates that have not been set, so it will create default certificates for all remaining certificates. It will not overwrite any custom certificates you have set. You can review the certificates prior to installation.
6. List all certificates with `apicup certs list SUBSYS`. The results will include the generated default certificates and the custom certificates that you set. Following is example output from the `apicup certs list` command:

Common certificates

=====

Name	Summary	Validation errors
analytics-ingestion-client	CN: analytics-ingestion-client SubjectKeyId: 27:60:BF:DF:6C:34:29:FE:8E:83:21:1B:C0:14:B2:9E AuthorityKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E	
ingress-ca	CN: ingress-ca SubjectKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E AuthorityKeyId: 5E:6D:5C:6E:2C:BE:50:F3:4E:EE:FD:02:76:86:6C:5A	
portal-client	CN: portal-client SubjectKeyId: 08:A8:57:A5:99:BC:79:FA:14:59:A4:98:6D:F7:43:C4 AuthorityKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E	
root-ca	CN: root-ca SubjectKeyId: 5E:6D:5C:6E:2C:BE:50:F3:4E:EE:FD:02:76:86:6C:5A AuthorityKeyId:	

Subsystem mgmt_subsys certificates

=====

Name	Summary	Validation errors
api-manager-ui	CN: api-manager-ui SubjectKeyId: F7:96:78:02:BE:01:83:C4:DF:42:A9:87:94:AB:1D:35 AuthorityKeyId: 6E:5A:6C:82:BA:F2:62:CF:B3:85:54:23:B6:26:9A:3F	
cloud-admin-ui	CN: cloud-admin-ui SubjectKeyId: AC:13:32:C2:6D:7B:46:6D:67:B5:41:6E:92:42:D3:4C AuthorityKeyId: 6E:5A:6C:82:BA:F2:62:CF:B3:85:54:23:B6:26:9A:3F	
consumer-api	CN: consumer-api SubjectKeyId: DE:84:86:40:4F:1E:30:A6:16:0E:CD:5B:8E:0C:1A:46 AuthorityKeyId: 6E:5A:6C:82:BA:F2:62:CF:B3:85:54:23:B6:26:9A:3F	
encryption-secret	A9:F3:28:0E:1E:AA:D9:5E:86:02:A4:95:69:83:94:30	
k8s-ca	CN: k8s-ca SubjectKeyId: 60:D9:B2:37:0B:17:FB:CD:FC:49:29:32:F6:A6:49:7C AuthorityKeyId:	
platform-api	CN: platform-api SubjectKeyId: AC:EF:88:78:01:A1:4D:8E:95:95:7D:3E:C5:43:F9:48 AuthorityKeyId: 6E:5A:6C:82:BA:F2:62:CF:B3:85:54:23:B6:26:9A:3F	

7. Install the subsystem with the certificates using `apicup subsys install`

SUBSYS. Any missing certificates will be generated. The installation will not proceed if there are any validation issues with the certificates. See [Validation reference](#).

8. Repeat for other subsystems.

9. If necessary, you can replace custom certificates after installation is complete. See [Replacing custom certificates](#).

Replacing custom certificates

Use the APICUP installer `certs` commands to replace existing certificates.

About this task

Important:

- Customization of public certificates and public user-facing certificates is recommended. Customization of internal certificates is strongly discouraged.
- To view a list of public, public user-facing, and internal certificates, see [Certificate management: Read This First](#). For details on each certificate, see [Certificate reference](#).

The APICUP installer can be used to update certificates for each subsystem after installation.

Requirements for custom certificates:

- Extended Key Usage (EKU), either `serverAuth` or `clientAuth` depending upon the type of certificate. Certificates of type *Server* must have an Extended Key Usage with `serverAuth` purpose. Certificates of type *Client* must have an Extended Key Usage with `clientAuth` purpose.
- Subject Alternative Name (SAN) for the required hosts
- Any custom common certificates that are being used must be set prior to setting any custom certificates for a subsystem.

See [Certificate Reference](#) to view the list of common certificates and to determine whether an EKU is needed for a certificate and which type of EKU (`serverAuth` or `clientAuth`).

Certificates and identical endpoints:

The Management subsystem has four public endpoints: `api-manager-ui`, `cloud-admin-ui`, `platform-api`, and `consumer-api`. Distinct TLS certificates can be set for each endpoint. However, if any two endpoints are identical, only one TLS certificate will be effective. When 2 or more endpoints are set to the same host, the secrets associated with the endpoints should be the same or contain the same certificate files.

Procedure

1. Generate the custom certificate with the appropriate EKU and SAN. You will need to obtain the private key, public certificate, and CA certificates in non-password-protected PEM format for the custom certificate. Following is an example for how to generate a certificate (`platform-api-example`) with an EKU `serverAuth` and SAN using `openssl`:

```
openssl x509 -req -days 360 -in platform-api-example.csr -CA root-ca.pem -CAkey root-ca-key.pem -CAcreateserial -out platform-api-cert -sha256 -extfile <(cat /etc/ssl/openssl.cnf <(printf "\n[SAN]\nsubjectAltName=DNS: fqdn.myserver.com\nextendedKeyUsage=serverAuth")) -extensions SAN
```

where

- `DNS:fqdn.myserver.com` is the fully qualified domain name of the endpoint the certificate applies to. This matches the endpoints entered in the APICUP installer.
- `platform-api-example.csr` is the file name for the certificate signing request

Following is an example for how to generate a certificate (portal-client) with an EKU clientAuth and SAN using openssl:

```
openssl x509 -req -days 360 -in portal-client-example.csr -CA root-ca.pem -CAkey root-ca-key.pem -CAcreateserial -out portal-client-cert -sha256 -extfile <(cat /etc/ssl/openssl.cnf <(printf "\n[SAN]\nkeyUsage=critical,digitalSignature,keyEncipherment\nextendedKeyUsage = clientAuth\nbasicConstraints=critical,CA:FALSE\nsubjectKeyIdentifier=hash\n")) -extensions SAN
```

2. Once the certificate has been created, set the certificate by entering the following command:

```
apicup certs set SUBSYS CERT_NAME [CERT_FILE KEY_FILE CA_FILE]
```

You can find definitions for commands at the following location: [Command reference](#)

If the certificate is signed by an intermediate CA, the `CA_File` argument must point to a file that concatenates the intermediate CA, followed by the root CA, in that order.

If the certificate was generated with an EKU serverAuth, it must be assigned to a server certificate. If the certificate was generated with an EKU clientAuth, it must be assigned to a client certificate.

3. Install the subsystem with the new certificate using `apicup subsys install SUBSYS`. Any missing certificates will be generated. The installation will not proceed if there are any validation issues with the certificates. See [Validation reference](#).
4. Repeat for other subsystems requiring new certificates.

Setting common certificates

Common certificates are set for one subsystem, but are applied to all subsystems. Use the APICUP installer `certs` commands to set the common certificates.

About this task

Important:

- Customization of public certificates and public user-facing certificates is recommended. Customization of internal certificates is strongly discouraged.
- To view a list of public, public user-facing, and internal certificates, see [Certificate management: Read This First](#). For details on each certificate, see [Certificate reference](#).

The common certificates are identical across subsystems. Subsystems require the common certificates to allow them to register with the management subsystem. When installing any one subsystem, the common certificates will be set for that subsystem and for all the other subsystems. If you are using custom certificates for the common certificates, they must be set prior to setting any custom certificates. See [Certificate reference](#) for a description of the common certificates.

Common certificates cannot be changed between subsystem installs. For example, you cannot set a common certificate for the management subsystem, install the management subsystem, then change a common certificate for the analytics subsystem, then install the analytics subsystem. This scenario will result in a failed installation because the common certificates are not identical.

Procedure

Follow these steps to set custom common certificates. If using default certificates, the common certificates will be set for you. See [Setting default certificates](#).

1. Set up and validate all subsystems. Enter the settings for the subsystem using `apicup subsys set <SUBSYS>` and validate the subsystem settings using `apicup subsys get <SUBSYS> --validate`. The subsystem must pass validation before setting the certificates.
2. Generate a custom certificate with the appropriate EKU and SAN. You will need to obtain the private key, public certificate, and CA certificates in non-password-protected PEM format for the custom certificate. Following is an example for how to generate a certificate (platform-api) with an EKU serverAuth and SAN using openssl:

```
openssl x509 -req -days 360 -in platform-api.csr -CA root-ca.pem -CAkey root-ca-key.pem -CAcreateserial -out platform-api-cert -sha256 -extfile <(cat /etc/ssl/openssl.cnf <(printf "\n[SAN]\nsubjectAltName=DNS:ac-msntest.myserver.com,DNS:ac2-msntest.myserver.com\nextendedKeyUsage=serverAuth")) -extensions SAN
```

Following is an example for how to generate a certificate (portal-client) with an EKU clientAuth and SAN using openssl:

```
openssl x509 -req -days 360 -in portal-client.csr -CA root-ca.pem -CAkey root-ca-key.pem -CAcreateserial -out portal-client-cert -sha256 -extfile <(cat /etc/ssl/openssl.cnf <(printf "\n[SAN]\nkeyUsage=critical,digitalSignature,keyEncipherment\nextendedKeyUsage = clientAuth\nbasicConstraints=critical,CA:FALSE\nsubjectKeyIdentifier=hash\n")) -extensions SAN
```

3. Once the certificate has been created and you have a `.pem` file, set the custom common certificates in one of your subsystems. After setting the custom certificates for one subsystem, they will take effect for all subsystems. The command is targeted at a specific subsystem, but the common certificates are copied to all subsystems regardless of which subsystem they are originally set in. Following is an example for setting the `portal-client` certificate:

```
apicup certs set mgmt portal-client myCertFile.pem myKeyFile.key myCAFile.crt
```
4. Set the remaining certificates for each subsystem, default or custom. See [Setting default certificates](#) and [Setting custom certificates](#).
5. List and validate the certificates for each subsystem. See [Validation reference](#).
6. Install each subsystem using `apicup subsys install SUBSYS`.

Setting the encryption-secret for the management database

Use the APICUP installer `certs` commands to set the encryption-secret for the management database.

About this task

The encryption-secret is a secure random bytes password used for field level encryption in the management database. You can generate 128 random bytes using the following command in openssl:

```
openssl rand -out /path/to/secret/encryption-secret.bin 128
```

Procedure

1. Enter the `apicup certs set SUBSYS CERT_NAME [KEY_FILE]` command and complete the following values:
 - SUBSYS - The subsystem for the `encryption-secret` is the name of your management subsystem, because it is used for field-level encryption for the management database.
 - CERT_NAME - The certificate name is `encryption-secret`.
 - KEY_FILE - Enter the file name for a secure random bytes string that is 128 bytes in length, for example `encryption-secret.bin`.
2. Set the remaining certificates if using custom certificates and install the management subsystem.

Clearing certificates

Certificates can be cleared in order to set new certificates.

About this task

Existing certificates must be cleared in order to set new certificates. When using default certificates, new certificates are created only for those certificates that are not set. Some of the use cases for clearing certificates are: expired certificates and configuration changes. For example, if endpoints are changed, the existing certificates must be cleared and new certificates created.

Procedure

1. Enter the `apicup certs set SUBSYS CERT_NAME --clear` command and complete the following values:
 - SUBSYS - The name of the subsystem
 - CERT_NAME - The name of the certificate that you want to clear.
2. The certificate will be cleared and can be reset, either as a default or custom certificate.

Reference for certificates, commands, and validations

This section contains the reference information for certificates, commands for working with certificates, and validations.

About this task

- [Certificate reference](#)
The Certificate reference provides a description of all the certificates required in API Connect.
- [Command reference](#)
The APICUP installer includes the `certs` commands to set and manage certificates.
- [Validation reference](#)
Certificates are validated using several parameters.

Certificate reference

The Certificate reference provides a description of all the certificates required in API Connect.

Before you begin

APICUP sets default certificates for each subsystem during installation. The default certificates are self-signed so may not provide a level of trust suitable for external communication. Custom certificates can be set and managed following the steps described in [Setting custom certificates](#).

The Certificates reference topic lists all certificates that are set by the `apicup certs` commands. The default certificates can be used for the majority of these certificates. The certificates that are marked as *public and user-facing* are recommended to be explicitly set as custom certificates because they are presented to an end user through a browser or Command Line Interface (CLI).

The certificates that are described as *TLS certificate used by ingress* are also considered *public* in the sense that they interact with a client that sits outside of an API Connect cluster.

Certificates that are listed as *auto-generated* cannot be set using the `apicup certs set` command. For example, the common certificate `ingress-ca` is auto-generated and used as an intermediate CA for all default ingress certificates. If you set an ingress certificate as a custom certificate, you will need to configure an intermediate CA if desired.

Important:

When setting custom certificates, additional steps must be taken to provide an Extended Key Usage (EKU) serverAuth and ClientAuth and a Subject Alternative Name (SAN) for the required hosts. These certificates are generated automatically by the `apicup certs` command when using the default certificates.

- For custom certificates of type *server*, an additional extended key usage client authentication (EKU serverAuth) certificate is required.
- For custom certificates of type *client*, an additional extended key usage server authentication (EKU clientAuth) certificate is required.

Procedure

1. **Common certificates** - The following certificates are common to all subsystems in a deployment. Subsystems require the common certificates to allow them to register with the management subsystem. The common certificates are identical across subsystems. When installing any one subsystem, the common certificates will be set for that subsystem and for all the other subsystems. Custom common certificates must be set prior to setting any custom subsystem certificates. Common certificates cannot be changed between subsystem installs. For example, you cannot set a common certificate for the management subsystem, install the management subsystem, then change a common certificate for the analytics subsystem, then install the analytics subsystem. This scenario will result in a failed installation because the common certificates are not identical.

Table 1. Common certificates

Certificate name (value used in <code>apicup certs</code>)	Usage	Requirements	Description
root-ca	CA		CA certificate which forms the root of the certificate chain
ingress-ca	CA	signed by: root-ca	Auto-generated intermediate certificate used to generate certificates for subsystem ingress endpoints if not provided by user. The ingress-ca intermediate certificate cannot be set explicitly, it is always only generated. TLS certificates for the ingresses are not required to use ingress-ca as an intermediate certificate, but if a given ingress TLS certificate remains to be auto-generated then it will be signed by this ingress-ca.
portal-client	Client	Must be signed by the same authority as the one used for the matching ingress TLS certificate <code>portal-admin-ingress</code> . For a two data center deployment, both data centers must have an identical subject name. For example, both data centers subject name could be <code>CN=portal-admin-client</code> , or they could both be <code>CN=ptl-adm-client, O=cert-manager</code> , but they must be identical.	Client certificate used by management subsystem to authenticate with the Portal admin endpoint. Requires EKU clientAuth.
analytics-client-client	Client	This is a legacy certificate, it is not used from v10.0.5 onwards	
analytics-ingestion-client	Client	Must be signed by the same authority as the one used for the matching ingress TLS certificate <code>analytics-ingestion-ingress</code> . For a two data center deployment, both data centers must have an identical subject name. For example, both data centers subject name could be <code>CN=a7s-ingestion-client</code> , or they could both be <code>CN=a7s-ingestion-client, O=cert-manager</code> , but they must be identical.	Client certificate used by management subsystem to authenticate with analytics client endpoint. Requires EKU clientAuth. Also used by gateway subsystem to authenticate with analytics ingestion endpoint.

2. **Management certificates**

These certificates apply to a single Management subsystem.

The Management subsystem has four public endpoints: `api-manager-ui`, `cloud-admin-ui`, `platform-api`, and `consumer-api`. Distinct TLS certificates can be set for each endpoint. However, if any two endpoints are identical, only one TLS certificate will be effective. When 2 or more endpoints are set to the same host, the secrets associated with the endpoints should be the same or contain the same certificate files.

The encryption-secret certificate is unique in that it is a secure random number. It is used to provide field level encryption in management (Cassandra) database. To set the encryption secret for the management database, use the following command: `apicup certs set SUBSYS CERT_NAME [KEY_FILE]` For example: `apicup certs set mgmt1 encryption-secret /path/to/keyfile`. See [Setting the encryption-secret for the management database](#)

Table 2. Management certificates

Certificate name	Usage	Requirements	Description
encryption-secret	secure random bytes	length: 128 bytes	Encryption secret used to do field level encryption in management (Cassandra) database
platform-api	Server	The host names for which the certificate is valid must include the platform-api endpoint. A wildcard certificate can be used as the first element in a host name, for example, if the endpoint is: <code>store.acme.com</code> , the certificate can be one that is valid for host names matching <code>*.acme.com</code> .	TLS certificate used by ingress. Requires EKU serverAuth. Public and user-facing.
consumer-api	Server	The host names for which the certificate is valid must include the consumer-api endpoint. A wildcard certificate can be used as the first element in a host name, for example, if the endpoint is: <code>store.acme.com</code> , the certificate can be one that is valid for host names matching <code>*.acme.com</code> .	TLS certificate used by ingress. Requires EKU serverAuth. Public and user-facing.

Certificate name	Usage	Requirements	Description
api-manager-ui	Server	The host names for which the certificate is valid must include the api-manager-ui endpoint. A wildcard certificate can be used as the first element in a host name, for example, if the endpoint is: <i>store.acme.com</i> , the certificate can be one that is valid for host names matching <i>*.acme.com</i> .	TLS certificate used by ingress. Requires EKU serverAuth. Public and user-facing.
cloud-admin-ui	Server	The host names for which the certificate is valid must include the cloud-admin-ui endpoint. A wildcard certificate can be used as the first element in a host name, for example, if the endpoint is: <i>store.acme.com</i> , the certificate can be one that is valid for host names matching <i>*.acme.com</i> .	TLS certificate used by ingress. Requires EKU serverAuth. Public and user-facing.
hub-endpoint	Server		Automated Testing Behavior UI and API endpoint
turnstile-endpoint	Server		Automated Testing Behavior UI and API endpoint

3. Portal certificates

These certificates apply to a single portal subsystem.

Table 3. Portal certificates

Certificate name	Usage	Requirements	Description
portal-admin-ingress	Server	host must match admin endpoint	TLS certificate used by ingress. The portal-client common certificate must be set prior to setting the portal-admin-ingress certificate. Requires EKU serverAuth. The portal-admin-ingress TLS certificate does not have any specific requirement on the authority signing it. If the certificate is auto-generated, it is signed by the ingress-ca .
portal-www-ingress	Server	host must match www endpoint	TLS certificate used by ingress. Requires EKU serverAuth. Public and user-facing.

4. Analytics certificates

These certificates apply to a single analytics subsystem.

Table 4. Analytics certificates

Certificate name	Usage	Requirements	Description
analytics-ingestion-ingress	Server	Required hosts: <ul style="list-style-type: none"> ingestion ingress endpoint *. <namespace> *. <namespace>.svc *. <namespace>.svc.cluster.local 	TLS certificate used by ingress. The analytics-ingestion-client common certificate must be set prior to setting the analytics-ingestion-ingress certificate. Requires EKU serverAuth. The analytics-ingestion-ingress TLS certificate does not have any specific requirement on the authority signing it. If the certificate is auto-generated, it is signed by the ingress-ca .

5. OVA/Appliance certificates

These certificates apply only to an appliance in a VMware environment. The **k8s-ca** and **appliance-client** certificates are auto-generated and cannot be set as custom certificates. These are used to set up the Kubernetes cluster in an OVA/Appliance.

Table 5. OVA/Appliance certificates

Certificate name	Usage	Requirements	Description
k8s-ca	CA		Auto-generated CA certificate that forms the root of the certificate chain for the Appliance/Kubernetes cluster components
appliance-client	Client	signed by: k8s-ca	Auto-generated client certificate used to communicate with the Appliance API server. Requires EKU clientAuth.

Command reference

The APICUP installer includes the **certs** commands to set and manage certificates.

About this task

The APICUP installer can be used to set certificates for each subsystem during installation. If certificates are not explicitly set using the **apicup certs set** command, then default certificates are generated by APICUP. We recommend that certificates be set at installation time only (or carried over from an upgrade). The default certificates are self-signed, so they may not be optimal for external communication.

For a description of the certificates that can be set, see [Certificate reference](#). We recommend that all public and user-facing certificates be explicitly set, including *portal-www-ingress* and *api-gateway-ingress*, and the four management endpoints (*platform-api*, *consumer-api*, *api-manager-ui*, and *cloud-admin-ui*). Following is the help reference for the **apicup certs set** command:

```
apicup certs set --help
Set or clear certificates and keys
```


Usage:
`apicup certs set SUBSYS CERT_NAME [CERT_FILE KEY_FILE CA_FILE] [KEY_FILE] [flags]`

Flags:
`--clear` Clear out a certificate or key entry
`-h, --help` help for set

Global Flags:
`--accept-license` Accept the license for API Connect
`--debug` Enable debug logging

Procedure

To set and clear certificates, complete the following steps:

1. Enter the `apicup certs set` command and complete the following values:

Table 1. `apicup certs set`

Command	Values	Result
<code>apicup certs set SUBSYS CERT_NAME [CERT_FILE KEY_FILE CA_FILE] [KEY_FILE] [flags]</code>	Parameters are: <ul style="list-style-type: none"> SUBSYS - name of the subsystem to which the certificate applies CERT_NAME - name of the certificate; see Certificate reference for a list of certificates that can be set for each subsystem. CERT_FILE - Path to the certificate file in PEM format. KEY_FILE - Path to the private key file in PEM format. CA_FILE - Path to the Certificate Authority (CA) file. The contents of the file may be the concatenation of an intermediate CA and the root CA (in that order). Note: When setting the <code>root-ca</code> certificate, omit the <code>CA_FILE</code> parameter. 	Applies the certificate when the subsystem is installed.
<code>apicup certs set SUBSYS CERT_NAME [KEY_FILE] [flags]</code>	<code>KEY_FILE</code> - The file containing the encryption-secret for field level encryption in the management database. Applies only to the management subsystem. The certificate name is <code>encryption-secret</code> . The type is secure random bytes with a length of 128 bytes. For example, <code>apicup certs set mgmt1 encryption-secret /path/to/encryption-secret.bin</code> . Note: Do not specify any of the <code>[CERT_FILE KEY_FILE CA_FILE]</code> parameters when setting the <code>encryption-secret</code> .	Applies the <code>encryption-secret</code> when the management subsystem is installed.
<code>flags</code> <ul style="list-style-type: none"> <code>--clear</code> <code>--help</code> 	Flags are: <ul style="list-style-type: none"> <code>--clear</code> - Clears the specified certificate. For example, <code>apicup certs set mgmt1 encryption-secret --clear</code> <code>--help</code> - Displays help for the command. 	The specified certificate will be cleared. When making configuration changes such as changing endpoints, the corresponding certificate must be cleared so that a new certificate can be set.

2. The `apicup certs get` command retrieves a specific certificate for the specified subsystem.

```
apicup certs get --help
Get a certificate
```

Usage:
`apicup certs get SUBSYS CERT_NAME [flags]`

Flags:
`-h, --help` help for get
`-o, --output string` output to file or - (stdout) (default "-")
`-t, --type string` type of object to return: cert, key, ca (default "cert")

Global Flags:
`--accept-license` Accept the license for API Connect
`--debug` Enable debug logging

Table 2. `apicup certs get`

Command	Values	Result
<code>apicup certs get SUBSYS CERT_NAME [flags]</code>	Parameters are: <ul style="list-style-type: none"> SUBSYS - name of the subsystem to which the certificate applies CERT_NAME - name of the certificate to retrieve; see Certificate reference for a list of certificates 	Returns the specified certificate for the specified subsystem.
<code>flags</code> <ul style="list-style-type: none"> <code>--output string</code> <code>--type string</code> <code>--help</code> 	Flags are: <ul style="list-style-type: none"> <code>--output string</code> - Specify a file for the retrieved values, or specify "-" to send to stdout. Default is "-" to send to stdout. For example, <code>apicup certs get mgmt1 --output myCertsFile</code> <code>--type string</code> - Returns only the specified type. If not specified, the type is <code>cert</code>. For example, <code>apicup certs get mgmt1 --type ca</code> <code>--help</code> - Displays help for the command. 	<ul style="list-style-type: none"> For <code>--output</code>: The specified certificate will be retrieved and sent to stdout or saved to the specified file For <code>--type</code>: Certificates will be retrieved that match the type specified.

3. List all certificates that have been set for a subsystem using the `apicup certs list` command. You can list the certificates at any time to summarize the certificates that have been set.

```
apicup certs list -help
```

List all configured certificates

Usage:
`apicup certs list SUBSYS [flags]`

Flags:
`-h, --help` help for list

Global Flags:

```
--accept-license  Accept the license for API Connect
--debug          Enable debug logging
```

Table 3. apicup certs list

Command	Values	Result
<code>apicup certs list SUBSYS [flags]</code>	Parameters are: <ul style="list-style-type: none"> SUBSYS - name of the subsystem for which you want to list certificates 	Returns a list of certificates that are configured for the subsystem.
<code>flags</code> <ul style="list-style-type: none"> <code>--help</code> 	Flags are: <ul style="list-style-type: none"> <code>--help</code> - Displays help for the command. 	Help text is displayed.

Following is example output from the `apicup certs list` command:

Common certificates

```
=====
```

Name	Summary	Validation errors
analytics-ingestion-client	CN: analytics-ingestion-client SubjectKeyId: 27:60:BF:DF:6C:34:29:FE:8E:83:21:1B:C0:14:B2:9E AuthorityKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E	
ingress-ca	CN: ingress-ca SubjectKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E AuthorityKeyId: 5E:6D:5C:6E:2C:BE:50:F3:4E:EE:FD:02:76:86:6C:5A	
portal-client	CN: portal-client SubjectKeyId: 08:A8:57:A5:99:BC:79:FA:14:59:A4:98:6D:F7:43:C4 AuthorityKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E	
root-ca	CN: root-ca SubjectKeyId: 5E:6D:5C:6E:2C:BE:50:F3:4E:EE:FD:02:76:86:6C:5A AuthorityKeyId:	

Subsystem mgmt certificates

```
=====
```

Name	Summary	Validation errors
api-manager-ui	CN: api-manager-ui SubjectKeyId: A3:C1:A1:4F:21:23:21:2F:1F:D7:87:30:E1:1E:33:A3 AuthorityKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E	
appliance-client	CN: appliance-client SubjectKeyId: 5C:FB:0F:5D:B8:BF:6F:89:CB:25:DD:54:31:A7:B4:63 AuthorityKeyId: 60:D9:B2:37:0B:17:FB:CD:FC:49:29:32:F6:A6:49:7C	
cloud-admin-ui	CN: cloud-admin-ui SubjectKeyId: E7:E2:D6:35:95:6B:D4:3B:F7:F7:9F:5F:DD:B8:02:E9 AuthorityKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E	
consumer-api	CN: consumer-api SubjectKeyId: 2A:80:EB:A6:31:9E:A5:C6:41:D9:1F:69:D1:9E:31:75 AuthorityKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E	
encryption-secret	2D:F9:61:0C:45:CB:6E:90:85:E0:0E:D3:DF:CC:B4:47	
k8s-ca	CN: k8s-ca SubjectKeyId: 60:D9:B2:37:0B:17:FB:CD:FC:49:29:32:F6:A6:49:7C AuthorityKeyId:	
platform-api	CN: platform-api SubjectKeyId: 6D:E7:60:21:81:7E:F6:40:A4:9A:2F:88:35:D1:18:04 AuthorityKeyId: 0B:37:61:5F:81:B3:67:5B:E0:F1:05:A6:6E:08:D5:8E	

4. The `apicup certs generate` command generates and sets default certificates. The `generate` command only generates and sets a certificate if it is not already set; it only sets the missing default certificates that have not been explicitly set using the `set` command. Execute the `generate` command before running the `apicup subsys install <SUBSYS>` command to confirm the certificates are correct before installing. It allows you to validate all certificates before performing the installation. The `generate` command is used as a tool to assist you when entering a combination of default and custom certificates. If you need to set specific certificates you can set them upfront (using `set`) and then generate the missing ones with default certificates. Or you can generate all certificates upfront and then override specific certificates to set custom certificates. Using `generate` helps to avoid validation errors during the installation procedure. Note that you must configure the subsystems and pass the `--validate` option before generating the default certificates.

```
apicup certs generate -help
Generate all unset certificates
```

```
Usage:
  apicup certs generate SUBSYS [flags]
```

```
Flags:
  -h, --help  help for generate
```

```
Global Flags:
  --accept-license  Accept the license for API Connect
  --debug          Enable debug logging
```

Table 4. apicup certs generate

Command	Values	Result
<code>apicup certs generate SUBSYS [flags]</code>	Parameters are: <ul style="list-style-type: none"> SUBSYS - name of the subsystem for which you want to generate certificates 	Generates certificates that have not been set for the subsystem. Generates self-signed certificates.
<code>flags</code> <ul style="list-style-type: none"> <code>--help</code> 	Flags are: <ul style="list-style-type: none"> <code>--help</code> - Displays help for the command. 	Help text is displayed.

Certificates are validated using several parameters.

The validations described in the following table are applied for all certificates, but are most helpful for custom certificates. For default certificates, the certificate validations will always pass, as the required elements are generated by APICUP. However, with custom certificates, some of the required elements may be missing or incorrect.

Validation	Messages	Error	See also/Action
Verify the certificate is set properly.	<code>certificate <cert> not set</code>	The certificate is not set.	
	<code>unable to load cert <cert></code>	The certificate is set but cannot be read.	
Verify certificate key usage (Extended Key Usage).	<code>unable to verify cert <cert>: missing key usage <n></code>	The certificate is missing the required key usage.	See Certificates Reference to see more information, including the type, for all certificates. See Setting custom certificates for tips on how to generate the EKUs for custom certificates.
Verify the certificate signing CA. If available, the CA file is loaded. Then the certificate is verified against the provided CA file, including enforcement of Extended Key Usage.	<code>unable to parse CA to verify cert <cert></code>	The CA file could not be parsed and loaded.	
	<code>unable to verify cert <cert></code>	The certificate failed verification against the provided CA file.	One possible reason for receiving this error is that the correct EKU is missing. For a custom certificate, see Setting custom certificates for information on generating EKUs.
Verify certificate hosts. The certificate must be valid for the hosts listed for the certificate in the Requirements column in the Certificates Reference.	<code>unable to verify cert <cert>: missing <host></code>	The certificate is not valid for the required host.	See Certificate reference for the required hosts.
Verify that a certificate that is being used as a CA is actually a CA.	<code>unable to verify cert <cert>: certificate is not a CA</code>	The certificate is not a valid CA.	
Verify client certificate match. The <code>portal-client</code> , and <code>analytics-ingestion-client</code> certificates are verified against the CA of, respectively, <code>portal-admin-ingress</code> , and <code>analytics-ingestion-ingress</code> .	<code>a CA certificate must be provided for this certificate</code>	The CA certificate is missing for one of the <code>portal-admin-ingress</code> , and <code>analytics-ingestion-ingress</code> .	The common certificates <code>portal-client</code> , and <code>analytics-ingestion-client</code> must be set prior to setting any custom certificates.
	<code>client cert cannot be verified against provided CA certificate</code>	The verification failed.	

Tips and tricks for using APICUP

The APICUP installer contains built-in time saving functions.

Introduction

This section describes tips and techniques for working with the APICUP installation commands. The APICUP installer creates charts and secrets that are then managed by Helm.

- [APICUP commands for the rotation of secrets on the Developer Portal subsystem](#)
- [Running Tiller with APICUP](#)
- [Entering multiple settings per command](#)
- [Entering multiple values](#)
- [Viewing the settings for a subsystem](#)
- [Validating the settings for a subsystem](#)
- [Output an installation plan](#)
- [Getting help for commands](#)
- [Getting help on a specific command](#)

APICUP commands for the rotation of secrets on the Developer Portal subsystem

To rotate the Developer Portal subsystem secrets, certificates, or both, run the following command:

```
apicup subsys rotate-secrets portal
```

Where:

- `portal` is the name of your Developer Portal subsystem.
- `--certs` (optional) is the list of certificates that you want to rotate. Listing an issuer will rotate all of the certificates that were issued by that issuer, for example, listing `portal-ca` will rotate the `portal-ca`, `portal-client`, and `portal-server` certificates.
- `--encryption-secret` (optional) is the value to set the encryption secret to, if the supplied secret exists in the environment. If not supplied, the rotated secret will be random.
- `--rotate-encryption-secret` (optional) determines whether the encryption secret is rotated. Can be set to `true` or `false`.
- `--wait` (optional) determines whether the command waits for the operation to complete or fail. Can be set to `true` or `false`.
- `--wait-timeout` (optional) is the duration in seconds of the command timeout. Default is 1m0s.

For example, to rotate the encryption secret to a random value run the following command:

```
apicup subsys rotate-secrets portal --rotate-encryption-secret true
```

The following example rotates the encryption secret to a named secret (note that first you must exec into the Developer Portal OVA to create the secret):

```
apicup subsys rotate-secrets portal --rotate-encryption-secret --encryption-secret my-new-secret
```

The following example rotates all of the Developer Portal certificates:

```
apicup subsys rotate-secrets portal --certs portal-ca
```

The following example rotates just the server and client certificates:

```
apicup subsys rotate-secrets portal --certs portal-server,portal-client
```

To list any Developer Portal secret rotations that were created by using the `rotate-secrets` command:

```
apicup subsys list-rotate-secrets portal --timeout 30
```

Where:

- `portal` is the name of your Developer Portal subsystem.
- `--timeout` (optional) is the duration in seconds of the command timeout. Default is 40 seconds.

To delete any Developer Portal secret rotations that were created by using the `rotate-secrets` command:

```
apicup subsys delete-rotate-secret portal --rotate-secret-name secret_name --wait true --wait-timeout 80
```

Where:

- `portal` is the name of your Developer Portal subsystem.
- `--rotate-secret-name` is the name of the Developer Portal secret rotation CR that you want to delete. You can find this name by running the `list-rotate-secrets` command.
- `--wait` (optional) determines whether the command waits for the operation to complete or fail. Can be set to `true` or `false`.
- `--wait-timeout` (optional) is the duration in seconds of the command timeout. Default is 1m0s.

Running Tiller with APICUP

To run Tiller in a namespace:

```
export TILLER_NAMESPACE=apic
```

Entering multiple settings per command

To configure multiple settings per command, enter a space between each setting and enter an equals sign (=) to configure the setting.

In the following examples, be sure to replace [spc] with an actual space.

You can set multiple database parameters on one line:

```
apicup subsys set mgmt cassandra-max-memory-gb=9[spc]cassandra-cluster-size=3[spc]cassandra-volume-size-gb=16[spc]cassandra-backup-protocol=sftp
```

You can set all endpoints on one line:

```
apicup subsys set mgmt platform-api=my.platform.com[spc]api-manager-ui=my.apim.com[spc]cloud-admin-ui=my.cloud.com[spc]consumer-api=my.consumer.com
```

Entering multiple values

Separate multiple values for a key/value pair with commas.

```
apicup subsys set mgmt dns-servers=8.8.8.8,4.4.4.4 search-domains=a.com,b.com
```

Viewing the settings for a subsystem

To view the current values that are set for a subsystem, enter the following command: `apicup subsys get <subsys_name>`. For example: `apicup subsys get mgmt` outputs the current values for the subsystem named `mgmt` and provides a description of each value. is organized into Kubernetes settings and Subsystem settings. Following is an example of the output from the `get` command

Figure 1. Output for the `get` command

```
Endpoints
=====
Name          Value          Description
-----
api-manager-ui  FQDN of API manager UI endpoint
cloud-admin-ui  FQDN of Cloud admin endpoint
consumer-api    FQDN of consumer API endpoint
platform-api    FQDN of platform API endpoint

Error: Subsystem validation failure. Run with --validate to see details
```

Validating the settings for a subsystem

The values set for a subsystem can be validated for syntax by entering the `--validate` option for the `get` command: `apicup subsys get <subsys_name> --validate`. For example: `apicup subsys get mgmt --validate` validates the current values for the subsystem named `mgmt`. In the following example of the output for the `--validate` option, the check mark indicates a valid setting. The x indicates an invalid setting with an error message provided.

Figure 2. Output for the `--validate` option

```
Endpoints
=====
Name                               Value
----                               -
api-manager-ui                      X api-manager-ui is not a valid hostname
cloud-admin-ui                      X cloud-admin-ui is not a valid hostname
consumer-api                         X consumer-api is not a valid hostname
platform-api                        X platform-api is not a valid hostname
```

Output an installation plan

Using APICUP, you can generate an installation plan and confirm it is correct prior to running the installation. You can then install the subsystem from the plan.

To output an installation plan, enter the following command:

```
apicup subsys install SUBSYS --out=install-plan
```

where `<install-plan>` is the name of the directory where the installation plan will be stored.

In the example, a directory named `install-plan` is created in the project directory. The `myProject/install-plan` directory contains the configuration parameters for the subsystem.

To install the subsystem from the install-plan, enter the following command from the project directory:

```
apicup subsys install SUBSYS --plan-dir=<full-path-to-plan-directory>
```

where `<full-path-to-plan-directory>` is the full qualified path to the plan directory.

Following are general rules for installing from the installation plan:

- Never edit the files in the output directory directly. Instead, if changes are needed, update the parameters using APICUP and generate a new plan.
- Always run APICUP commands from the original project directory created during the initial product installation. The project directory contains the `apiconnect-up.yml` file.
- You can generate the plan in any location, but the `install` command must be run from the project directory. Enter the full path to the plan as the argument to `--plan-dir` to perform an installation.

The plan must be current with the project and the certificates. If the plan is older than the last modification date of the project or certificates, you will receive an error message such as:

```
the project was modified since the plan was generated, regenerate plan or skip this check
with a --skip-health-check flag in the command
```

or

```
the certs were changed since the plan was generated, regenerate plan or skip this check with
a --skip-health-check flag in the command
```

Getting help for commands

You can get help for all commands by entering: `apicup --help`. Following is an example of the output:

Figure 3. Help for `apicup` commands

```
APIConnect Install Assist
Usage:
  apicup [command]

Available Commands:
  certs           Subsystem certificates
  completion     Generates bash or zsh completion scripts
  help           Help about any command
  hosts          Commands to configure subsystem hosts
  iface          Commands to configure hosts interfaces
  init           Create a new APIConnect UP project
  registry-upload Retag and upload images to custom registry
  server         Starts the APIConnect cluster operator
  subsys         Subsystem commands
  version        Get the APIConnect UP version

Flags:
  --accept-license  Accept the license for API Connect
  --debug           Enable debug logging
  -h, --help       help for apicup

Use "apicup [command] --help" for more information about a command.
```

Getting help on a specific command

For help on a specific command, enter `--help` after the command. For example, `apicup subsys get mgmt --help` prints out the usage and flags for the `get` command. For example:

Figure 4. Help for get command

```
Get subsystem properties

Usage:
  apicup subsys get SUBSYS [flags]

Flags:
  --endpoints      List endpoints (default true)
  -h, --help       help for get
  --platform       List platform settings (default true)
  --subsystem      List subsystem settings (default true)
  --validate       Validate settings

Global Flags:
  --accept-license  Accept the license for API Connect
  --debug           Enable debug logging
```

Installing API Connect

Use these instructions to install or upgrade a deployment of API Connect on VMware.

- [IBM API Connect Version 10 software product compatibility requirements](#)
Ensure that you install the minimum API Connect operating system requirements. Use the IBM® Software Product Compatibility Reports site to generate a requirements report appropriate for your API Connect version and environment.
- [Requirements for initial deployment on VMware](#)
Review the requirements and considerations for deploying in a VMware environment.
- [API Connect licenses](#)
When you install a new version of API Connect on VMware, you must accept a new license.
- [What's new for v10 installation](#)
Summary of installation changes for API Connect v10 on VMware.
- [First steps for deploying in a VMware environment](#)
The first steps in deploying in a VMware environment are to obtain the API Connect distribution files and to create a project directory.
- [Signature verification by using PGP](#)
You can verify the integrity of files to ensure that they originated from IBM and are not modified.
- [Deploying the Management virtual appliance](#)
You can create a virtual server by deploying the relevant IBM API Connect OVA file on a VMware virtual server. Create all of the virtual servers that you want to use in your cloud.
- [Deploying the Analytics virtual appliance](#)
Install the IBM API Connect Analytics subsystem.
- [Deploying the Developer Portal virtual appliance](#)
You create a Developer Portal node by deploying the Developer Portal OVA template. After you deploy the Developer Portal OVA template, you can install the Developer Portal.
- [Deploying DataPower Gateway virtual appliance](#)
API Connect uses IBM DataPower® Gateway to provide the gateway service.
- [Post-deployment steps](#)
When all subsystems are deployed, use the admin account to access the Cloud Manager administrative console and begin configuration. Optionally, take VMware snapshots of the VMs.
- [Installing a two data center deployment on VMware](#)
How to install a two data center disaster recovery (2DCDR) deployment on VMware.

IBM API Connect Version 10 software product compatibility requirements

Ensure that you install the minimum API Connect operating system requirements. Use the IBM® Software Product Compatibility Reports site to generate a requirements report appropriate for your API Connect version and environment.

Generating a Software Product Compatibility Report

To generate an API Connect requirements report, complete the following steps:

1. Open the [Detailed system requirements for a specific product](#) page on the IBM Software Product Compatibility Reports site.
2. Search for the IBM API Connect product.
3. In the Search results list, select IBM API Connect.
4. From the Version list, select the required version.
5. Use the Filters to refine the contents of the requirements report.
6. Click Submit to generate your requirements report.

Important: When reviewing the report to determine your deployment needs, you should treat the stated CPU and memory requirements as reserved VMware capacity. For example, if the system requirements are 4 CPU and 16GB RAM, then those resources should be available at all times from the underlying hardware and should not be coming from an over-committed pool of resources.

Requirements for initial deployment on VMware

Review the requirements and considerations for deploying in a VMware environment.

- [Deployment requirements on VMware](#)
- [Disk space requirements](#)
- [API Connect configuration on VMware](#)
- [Passwords and certificates](#)
- [Setting and using a hashed default password](#)
- [DataPower Gateway for API Connect on VMware](#)

Deployment requirements on VMware

- Ensure you have supported software requirement versions. See [IBM API Connect Version 10 software product compatibility requirements](#). Attention: API Connect is not supported on a FIPS-enabled environment.
- Do not change the hardware version of the OVA during installation. Do not attempt to use an unsupported version, even if VMware indicates compatibility with other versions. For example, when deploying API Connect, the VMware UI for the APIC OVAs might show information like:

Table 1.

Property	Value
Guest OS	Ubuntu Linux (64-bit)
Compatibility	ESXi 5.5 and later (VM version 10)
VMware Tools	Yes
CPUs	4
Memory	16 GB

Although the **Compatibility** field shows **ESXi 5.5 and later (VM version 10)**, API Connect supports only the versions listed in [IBM API Connect Version 10 software product compatibility requirements](#). Do not change the VM version of the OVA. Ensure that the **Compatibility** field values are not changed, and remain **ESXi 5.5 and later (VM version 10)**.

Attempts to modify the VMware compatibility may typically result in failure to boot the OVA, as per <https://kb.vmware.com/s/article/52683>.

- Ensure that your operating system has one of the supported utilities for creating ISOs. The apicup installer uses **mkisofs** on Linux, and **hdiutil** on macOS. For Windows, you need software that creates ISO files using **mkisofs**, such as CDRTools. Verify that the utility you will use is located in a directory that is referenced by the PATH environment setting for your operating system. When creating the ISO, if you encounter the message **Error: unable to create config ISO for host**, verify that you have sufficient permissions to run the command.
- **Important:** Use a single **apicup** project for all subsystems, even those in a different cluster. Multiple projects will result in multiple certificate chains which will not match.

The original project directory created with **apicup** during the initial product installation (for example, *myProject*) is required to both restore the database and to upgrade your deployment. You cannot restore the database or perform an upgrade without the initial project directory because it contains pertinent information about the cluster. Note that the endpoints and certificates cannot change; the same endpoints and certificates will be used in the restored or upgraded system. A good practice is to back up the original project directory to a location from where it can always be retrieved.

Endpoints for the components cannot change between deployments. However, the endpoints for the VMware hosts can be modified for the new deployment.

- For each subsystem, gather the following networking settings, which you will need to supply during configuration:

Table 2.

Required information	Value for your system
IP address of the server	
IP address of the server	
Domain of the server	
IP addresses of the name servers	
IP address of the network gateway (not DataPower® gateway) for the server	
Name of the Ethernet interface	
VLAN	

Some virtualization environments require additional information when you create and configure virtual machines. For example, it might be necessary to assign a specific VLAN ID, Resource Pool, or Datastore. Please refer to information provided by your virtualization environment administrators.

Disk space requirements

- For Version 10, the data disk requirement for the management server is 200 GB. The boot disk requires 100 GB, so a minimum of 300 GB is needed for the management subsystem. You specify the data disk size in [Deploying the Management subsystem OVA file](#) when you deploy the OVF template in the VMware vSphere Web Client.
- The Developer Portal disk requirements vary with usage scenarios. See [Table 1](#).
- Before you install, review all the minimum hardware requirements listed on the Hardware tab in the Software Product on the Compatibility Reports for API Connect. Follow the instructions in [IBM API Connect Version 10 software product compatibility requirements](#).

API Connect configuration on VMware

- API Connect cannot be deployed on NFS.
- The timezone for API Connect pods is set to UTC. In API Connect deployments on VMware, the operating system timezone is also set to UTC. Do not change the timezone for the pods or the operating system.
- Ensure that the time settings match (within a few seconds) between the machine running **apicup** and the VMware **Host** clock. To verify the VMware host clock setting, see <https://kb.vmware.com/s/article/1003736>.

If the clocks are too different, the installation can fail because of invalid certificates due to time discrepancies.

- Deploying multiple similar subsystems in a single namespace is not supported. For example, deploying two management subsystems, `mgmt1`, `mgmt2` in a single namespace where `apiconnect operator` is installed.
- API Connect requires a dedicated IP range for deployment of the Kubernetes pod and Kubernetes service networks. These IP addresses cannot conflict with IP addresses used by other resources in your deployment, such as SMTP servers or user registries. The default values are 172.16.0.0/16 and 172.17.0.0/16, respectively. If a /16 subnet overlaps with existing IPs on the network, a CIDR as small as /22 is acceptable. If the default ranges conflict with other programs, you can modify the API Connect ranges during initial installation. Note that you cannot modify them once an appliance has been deployed. Follow the instructions for your subsystem if you want to modify either of the IP ranges.

- Only static IP addresses that are specified during the `apicup` project configuration before the installation of the OVAs are supported.
- Designated host names must have wildcard aliases or host aliases, which ensures that the different endpoints work together. For example, `*.hostname.mycompany.com`.
- Endpoints cannot contain the underscore character `"_"` or uppercase characters.
- Host names that are used to create endpoints cannot contain the underscore character `"_"` or uppercase letters.
- Kubernetes ingress limits the character set for DNS names to not support the underscore character `"_"`. This means you cannot specify underscores in domain names that are used as endpoints. For example, `my_domain.bar.com` and `my.domain_abc.com` are not supported for `<xxx>.<hostname>.<domainname>`, and will cause an error. For example:

```
Invalid value: "my_domain.bar.com": a DNS-1123 subdomain must consist of lowercase alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character (e.g. 'example.com', regex used for validation is '[a-z0-9]([-a-z0-9]*[a-z0-9])?(\.[a-z0-9]([-a-z0-9]*[a-z0-9])?)*')
```

- Note that you cannot change host names or DNS names on a running cluster.
- For Analytics, the installer specifies a default value of 16 GB for `es-max-memory-gb`. The minimum value for this setting is 12 GB.
- Decide the `deployment-profile` to use for the installation. See [Planning your deployment topology and profiles](#). Use of three replica deployment-profile is supported only on installations with three or more nodes. Installations with less than three nodes must use a one replica deployment-profile.

Note that a deployment-profile is set for each subsystem type: Management, Analytics, and Developer Portal.

- For additional considerations for configuring clusters, see [Configuring API Connect subsystems in a cluster](#). Important: For best performance it is recommended that the network latency between any 2 nodes be as low as possible. Do not configure nodes from the same subsystem cluster across multiple data centers with a high latency network. A high latency network is one that experiences more than 30ms latency between nodes.
- Configure a remote server for logging. Follow these instructions: [Configuring remote logging for a VMware deployment](#). Pods are terminated during an upgrade, and the logs will be lost if not stored remotely.

Passwords and certificates

- When creating configuration files for use in generating an ISO image for VMware, ensure that your working directories are secure. The VMware configuration requires an ISO image that contains a plain text password to be used to unlock the VMware data disk. This means that the API Connect project configuration file `apiconnect-up-v10.yml`, and the `/user-data` directory for each host, contain the passwords you specified. This configuration information is used to create the ISO image that you combine with the `.ova` distribution files when deploying (unlocking) the VMware data disk.
- You can use `apicup` to specify an `ssh` keyfile that contains a public certificate for using `ssh` to log in to a virtual machine. Logging in through `ssh` is preferred because it is more secure than password-based login.
- Default certificates are generated for each subsystem by the `apicup subsys install` command. If certificates are not explicitly set using the `apicup certs set` command, then default certificates are automatically generated by `apicup`. The default certificates are self-signed, so they may not provide a level of trust suitable for external communication. See [Working with certificates](#).

Setting and using a hashed default password

During configuration of the Management, Analytics, and Developer Portal subsystems, you create a password to use to log in to the management console for the first time. You must use a password hashing utility to hash the password. You then use `apicup` to assign this hashed password to the subsystem. These configuration steps ensure that the password is not stored in plain text on the data disk.

The syntax of the `apicup` command is:

```
apicup subsys set mgmt default-password='hashed_password'
```

Usage notes:

- The `default-password` is for the `apicadm` user account on the Appliance.
- The password for `apicadm` can be used **only** to log in through the VMware console. You cannot use it to `ssh` into the Appliance as an alternative to using the `ssh-keyfiles`. Interactive login for `ssh` is disabled.
- The `default-password` value configured is only used during initial installation (first boot) of each virtual appliance. Changing the value, then regenerating the ISO, and attaching the new ISO to the virtual appliance does not change the `apicadm` password.
- The `default-password` must be hashed. If it is plain text, you will not be able to log into the Appliance through the VMware console. When you use `apicup` to set or get `default-password`, `apicup` ensures that the hash type of the password is one of the following:
 - MD5
 - SHA1
 - SHA256
 - SHA512
 - BCRYPT
 - MD5-Crypt
- When using `apicup` to set a default password for a subsystem, be aware of syntax differences between operating systems. Windows requires double-quotes. Linux and OSX require single quotes.

Operating system	Command syntax
Linux or OSX	<code>apicup subsys set mgmt default-password='hashed_password'</code>
Windows	<code>apicup subsys set mgmt default-password="hashed_password"</code>

- You can use the `passwd` command (on the appliance) to change the `apicadm` password.

- When using the VMware Remote console to login to the appliance, be aware that the keyboard layout is English. This can cause a problems with hashed passwords, if you created the ISO on a system with a different keyboard layout and you used special characters or symbols. Passwords are hashed when you set the password that you enter to log into your Management appliance for the first time, and when you create hosts.

DataPower Gateway for API Connect on VMware

- Installation and configuration of DataPower Gateway on an appliance (physical or virtual) is completed after you install the API Connect subsystems. For the gateway service in a VMware environment, use the instructions in [Deploying DataPower Gateway virtual appliance](#).
- Ensure that DataPower Gateway firmware version you plan to install is compatible with the API Connect Management server version.

Note:

You can use any combination of API Connect 10.0.5.x with DataPower Gateway 10.5.0.x or DataPower API Gateway 10.5.0.x.

Before you install, best practice is to review the latest compatibility support for your version of API Connect. To view compatibility support, follow the instructions in [IBM API Connect Version 10 software product compatibility requirements](#) to access API Connect information on the Software Product Compatibility Reports website. Once you access information for your version of API Connect, select Supported Software_>, Integration Middleware, and view the list of compatible DataPower Gateway versions.

API Connect licenses

When you install a new version of API Connect on VMware, you must accept a new license.

Setting the License ID on VMware

Each license has a unique identifier called a **License ID**. Use **apicup** to accept the license. License acceptance is stored in **apiconnect-up-v10.yml**. Each accepted **License ID** is passed into subsystem CRs.

```
apicup licenses accept <License_ID>
```

For example:

```
apicup licenses accept L-EMZJ-MQ4M24
```

- Use **apicup licenses list** displays licenses available for current version. Also shows all accepted licenses, including prior versions. **apicup licenses list** has a **-o yaml|json** flag to make script based usage easier.
- If an invalid value is passed to **apicup licenses accept**, an error is displayed showing the valid values:

```
Error: Must run `apicup licenses accept ID` with one of [L-GVEN-GFUPVE, L-BYRR-EEASJ3, .....]
```

- If you run other **apicup** commands, such as **apicup subsys get <subsystem_name>** from within a project directory, without first accepting a license, those commands will fail. Once you accept the license, the commands will succeed.

You accept the license during [First steps for deploying in a VMware environment](#).

API Connect 10.0.5 licenses

For API Connect 10.0.5.3 and later, the licenses in Table 1 will be used as the master mod-level license for all Fix Pack and security roll-up releases in this stream. There are no changes to license terms, only updates to open-source notices documented in the notes for each.

Table 1. API Connect 10.0.5 licenses

License ID	Program Name	URL
L-VQYA-YNM22H	IBM API Connect Enterprise V10.0.5.3	https://ibm.biz/BdPGTL
L-SLCD-D9MBTA	IBM API Connect Enterprise Add-on for IBM Products V10.0.5.3	https://ibm.biz/BdPGTQ
L-XSJQ-UMJP6P	IBM API Connect Professional V10.0.5.3	https://ibm.biz/BdPGTC

Table 2. API Connect 10.0.5.2 licenses

License ID	Program Name	URL
L-GVEN-GFUPVE	IBM API Connect Enterprise V10.0.5.2	https://ibm.biz/BdPPmt
L-BYRR-EEASJ3	IBM API Connect Enterprise Add-on for IBM Products V10.0.5.2	https://ibm.biz/BdPPm6
L-EMZJ-MQ4M24	IBM API Connect Professional V10.0.5.2	https://ibm.biz/BdPPmU

Table 3. API Connect 10.0.5.0 and 10.0.5.1 licenses

License ID	Program Name	URL
L-RJON-CEBLEH	IBM API Connect Enterprise PVU V10.0.5	https://ibm.biz/BdPFpN
L-RJON-CEBLCF	IBM API Connect Enterprise PVU Add-on for IBM Products V10.0.5	https://ibm.biz/BdPFpW
L-RJON-CEBL97	IBM API Connect Professional PVU V10.0.5	https://ibm.biz/BdPF8B

What's new for v10 installation

Summary of installation changes for API Connect v10 on VMware.

- Changes to **apicup** settings:
 - **license-use**. Values changed: **Production** is now **production**. **Nonproduction** is now **nonproduction**.
 - **licenses accept** added for setting Program license. See [API Connect licenses](#)

- **mode** is deprecated and replaced by **deployment-profile**. Each subsystem (Management, Analytics, Portal) has separate values for one replica profiles and three replica profiles.
- v2018 settings that are deprecated:
 - **az-name** - no longer used with the Management subsystem
 - **cross-az-peers** - no longer used with the Management subsystem
 - **cassandra-max-memory-gb** - no longer used with the Management subsystem
 - **es-memory-gb** - no longer used with the Analytics subsystem.
 - All **cassandra-backup-*** settings are no longer used with the Management subsystem. The Cassandra database is deprecated and replaced by Postgres.
 - **cassandra-backup-auth-pass**
 - **cassandra-backup-auth-user**
 - **cassandra-backup-host**
 - **cassandra-backup-path**
 - **cassandra-backup-port**
 - **cassandra-backup-protocol**
 - **cassandra-backup-schedule**
- The minimum size data disk for the Management system increases from 150 GB in v2018 to 200 GB in v10. When combined with the 100 GB requirement for the root disk (unchanged), you need 300 GB for the Management subsystem.
- Upon reboot, any extra disk that was added to the VM is picked up and added to the secure storage (data disk).

First steps for deploying in a VMware environment

The first steps in deploying in a VMware environment are to obtain the API Connect distribution files and to create a project directory.

Before you begin

- Review [Requirements for initial deployment on VMware](#).
- See [What's new for v10 installation](#)
- See also [Configuring API Connect subsystems in a cluster](#).

About this task

In these First Steps, you will use the **apicup** command to create a project directory that contains the configuration file **apiconnect-up-v10.yml**. When you configure the first subsystem, such as the Management Service, the subsystem configuration settings are placed in the file.

When you want to configure a second subsystem, it is important that you reuse the same project directory. This means that you only need to download **apicup** once, and create a project directory once. The configuration settings for the second subsystem are added to the existing **apiconnect-up-v10.yml**. In this way, the second subsystem, such as Developer Portal, can access the configuration information needed to interact with the first subsystem.

After completing these First Steps, follow the links to instructions for configuring your subsystem. You will specify configuration settings, create an ISO of the subsystem configuration, and then use VMware to deploy the distribution file (.ova file) for the subsystem with your ISO. Note that you create a separate ISO for each subsystem.

Installation and configuration of DataPower® Gateway on an appliance (physical or virtual) is completed after you install the API Connect subsystems. For the gateway service in a VMware environment, use the instructions in [Deploying DataPower Gateway virtual appliance](#).

Note:

- For upgrades, do not use the procedures on this page. See [Upgrading API Connect](#).
- Some **apicup** commands have changed from previous versions of API Connect. See [What's new for v10 installation](#).
- When maintaining API Connect, do not use **kubect1 exec** commands to access API Connect pods unless advised by IBM. Do not make any changes on the deployed VMs unless documented here or otherwise advised by IBM. Attempting to manually update packages, adding new users, or installing new software will likely cause problems. Operating system updates are handled by API Connect fix packs.

Procedure

1. Obtain the API Connect files.

The following files are used for initial deployment on VMware:

IBM® API Connect <version> Management for VMware
Management subsystem files

IBM® API Connect <version> Developer Portal for VMware
Developer Portal subsystem files

IBM® API Connect <version> Analytics for VMware
Analytics subsystem files

IBM® API Connect <version> Install Assist (apicup) for <operating_system_type>
The apicup installation utility. Required for all installations on VMware.

IBM® API Connect <version> Toolkit for <operating_system_type>

Toolkit command line utility. Packaged standalone, or with API Designer or Loopback:

- IBM® API Connect <version> Toolkit for <operating_system_type>
- IBM® API Connect <version> Toolkit with Loopback for <operating_system_type>
- IBM® API Connect <version> Toolkit Designer with Loopback for <operating_system_type>

Not required during initial installation. After installation, you can download directly from the Cloud Manager UI and API Manager UI. See [Installing the toolkit](#).

IBM® API Connect <version> Local Test Environment

Optional test environment. See [Testing an API with the Local Test Environment](#)

IBM® API Connect <version> Security Signature Bundle File

Checksum files that you can use to verify the integrity of your downloads.

Note: The Fix Pack page contains two copies of the files for each subsystem. One copy is for new installations and one is for upgrades of existing installations. For new installations, do not use files with *Upgrade* in the file name.

2. Select one of the following actions:

- If you are configuring your first subsystem, go to step [3](#).
- If you have already configured a subsystem, go to step [5](#).

3. Install the installation utility.

- a. Locate the apicup installation utility file for your operating system, and download it:
- b. Rename the file for your OS type to **apicup**. Note that the instructions in this documentation refer to **apicup**.
- c. OSX and Linux® only: Make the apicup file an executable file by entering the following command:

```
chmod +x apicup
```

d. Set your path to the location of your apicup file.

- For the OSX and Linux operating systems:

```
export PATH=$PATH:/Users/your_path/
```

- For the Windows operating system:

```
set PATH=c:\your_path;%PATH%
```

4. Create a project called *myProject* and copy the **apicup** executable into the project directory:

a. Create a project:

```
apicup init myProject
```

b. Copy the **apicup** executable into the *myProject* directory or folder created by the **apicup init** command.

c. From within your project directory, specify the API Connect license:

```
apicup licenses accept <License_ID>
```

The **<License_ID>** is specific to the API Connect Program Name you purchased. To view the supported license IDs, see [API Connect licenses](#).

Important:

Use a single APICUP project for all subsystems, even those in a different cluster. Multiple projects will result in multiple certificate chains which will not match.

The original project directory created with APICUP during the initial product installation (for example, *myProject*) is required to both restore the database and to upgrade your deployment. You cannot restore the database or perform an upgrade without the initial project directory because it contains pertinent information about the cluster. Note that the endpoints and certificates cannot change; the same endpoints and certificates will be used in the restored or upgraded system. A good practice is to backup the original project directory to a location from where it can always be retrieved.

Endpoints for the components cannot change between deployments. However, the endpoints for the VMware hosts can be modified for the new deployment.

The apiconnect-up-v10.yml file is created in that directory.

Important: The apiconnect-up-v10.yml file must be in a secure and permanent location. The file contains password information and other information that is exposed in text format. Ensure that the project directory is secure.

5. Continue with the configuration steps for your subsystem:

- [Deploying the Management virtual appliance](#)
- [Deploying the Analytics subsystem](#)
- [Deploying the Developer Portal virtual appliance](#)

Signature verification by using PGP

You can verify the integrity of files to ensure that they originated from IBM and are not modified.

About this task

You can use code signatures to verify that a downloaded file was created by IBM and that no bits in the file were changed. The signature files that are used are *.asc. The key/cert file is PRD0003216key.pub.pgp.

Procedure

1. Import the public key.

```
$ gpg --import PRD0003216key.pub.pgp
gpg: key 020ED6B5DBE65F3B: public key "APIConnect <psirt@us.ibm.com>" imported
gpg: Total number processed: 1
gpg:          imported: 1

$ gpg --list-key --fingerprint APIConnect
pub   rsa4096 2023-02-28 [SCE]
      39ED 1634 5FCD EDB5 D6ED  E860 020E D6B5 DBE6 5F3B
uid           [ unknown] APIConnect <psirt@us.ibm.com>
```

When the key is imported, it does not need to be imported again. It is not signed, but you should make a note of the fingerprint so you can compare to make sure that the key is IBM's:

```
39ED 1634 5FCD EDB5 D6ED  E860 020E D6B5 DBE6 5F3B
```

2. Verify the files.

```
$ $ gpg --verify helper_files.zip.asc
gpg: assuming signed data in 'helper_files.zip'
```

```
gpg: Signature made Mon 28 Sep 17:35:17 2020 PDT
gpg:                using RSA key 39ED16345FCDEDB5D6EDE860020ED6B5DBE65F3B
gpg: Good signature from "APIConnect <psirt@us.ibm.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:                There is no indication that the signature belongs to the owner.
Primary key fingerprint: 39ED 1634 5FCD EDB5 D6ED E860 020E D6B5 DBE6 5F3B
```

The verification shows that the key matches the signature and that shows the fingerprint of the key, which should match the import:

```
39ED 1634 5FCD EDB5 D6ED E860 020E D6B5 DBE6 5F3B
```

Deploying the Management virtual appliance

You can create a virtual server by deploying the relevant IBM® API Connect OVA file on a VMWare virtual server. Create all of the virtual servers that you want to use in your cloud.

Before you begin

Before you deploy:

- Review the [Deployment requirements on VMware](#).
- Review the [API Connect configuration on VMware](#).
- Review [What's new for v10 installation](#)
- For information on deploying a cluster, see [Configuring API Connect subsystems in a cluster](#)
- If you are upgrading from a previous version, see [Upgrading API Connect](#).

About this task

You must deploy the API Connect OVA template to create each Management virtual server that you want in your cloud.

Attention: The following settings apply to features that are currently not supported, and should not be modified:

- Subsystem settings: test-and-monitor-enabled
- Certificate settings: hub, turnstile

Procedure

Complete each of the following tasks in order:

1. [Configuring the Management subsystem](#)
2. [Deploying the Management subsystem OVA file](#)
3. [Verify installation of the Management subsystem](#)

What to do next

If you want to deploy an API Connect Analytics OVA file, continue with [Deploying the Analytics subsystem](#).

If you want to deploy an API Connect Developer Portal OVA file, continue with [Deploying the Developer Portal virtual appliance](#).

Identify the DataPower® appliances to be used as gateway servers in the API Connect cloud and obtain the IP addresses.

- [Configuring the Management subsystem](#)
Specify configuration properties for your management subsystem, and create an ISO.
- [Deploying the Management subsystem OVA file](#)
After you have specified configuration settings and created an ISO, deploy the subsystem OVA into an appliance VM.
- [Verify installation of the Management subsystem](#)
Verify that deployment of the management subsystem OVA file succeeded.

Configuring the Management subsystem

Specify configuration properties for your management subsystem, and create an ISO.

Before you begin

Review [Deploying the Management virtual appliance](#)

About this task

Use the `apicup` installation utility to specify configuration settings for your management subsystem.

Procedure

1. Ensure that you obtained the distribution file and have a project directory, as described in [First steps for deploying in a VMware environment](#).

2. Change to the project directory.

```
cd myProject
```

3. Create a management subsystem.

```
apicup subsys create mgmt management
```

Where:

- `mgmt` is the name of your management server that you are creating. You can assign it any name, as long as the identifier consists of lowercase alphanumeric characters or '-', with no spaces, starts with an alphabetic character, and ends with an alphanumeric character.
- `management` indicates that you are creating a management microservice.

The API Connect Helm charts are deployed into the default namespace. You do not need to specify a namespace.

Tip: At any time, you can view the current management subsystem values in the `apiconnect-up-v10.yml` by running the `apicup subsys get` command:

```
apicup subsys get mgmt
```

If you have not yet configured the subsystem, the command might return errors. Also, if you have not updated the value, a default value is listed, if there is one that is available.

After configuration is complete, you can view output similar to the following sample:

```
Appliance settings
=====
Name                               Value
Description                         -----
-----
additional-cloud-init-file
(Optional) Path to additional cloud-init yml file
data-device                          sdb
VM disk device (usually `sdb` for SCSI or `vdb` for VirtIO)
default-password
$6$rounds=4096$vtcqpAVK$dzqrOeYP33WTvTug38Q4Rld5l8TmdQgezZTnkX/PFwkzTiZ2S0CqNRr1S4b08tOc4p.OEg4BtzBe/r8RAk.gw/
(Optional) Console login password for `apicadm` user, password must be pre-hashed
dns-servers                          [8.8.8.8]
List of DNS servers
extra-values-file
(Optional) Path to additional configuration yml file
k8s-pod-network                      172.16.0.0/16
(Optional) CIDR for pods within the appliance
k8s-service-network                 172.17.0.0/16
(Optional) CIDR for services within the appliance
public-iface                         eth0
Device for API/UI traffic (Eg: eth0)
search-domain                       [subnet1.example.com]
List for DNS search domains
ssh-keyfiles                        [/home/vsphere/.ssh/id_rsa.pub]
List of SSH public keys files
traffic-iface                       eth0
Device for cluster traffic (Eg: eth0)

Subsystem settings
=====
Name                               Value
Description                         -----
-----
deployment-profile                  nlxc4.m16
Deployment profile (nlxc2.m16/n3xc4.m16) for analytics, (nlxc4.m16/n3xc4.m16) for management, (nlxc2.m8/n3xc4.m8) for portal
license-use                         production
License use (production/nonproduction)
multi-site-ha-enabled              false
Multi site HA enabled
multi-site-ha-mode                 active
Multi site HA mode (active/passive)
replication-peer-fqdn
Replication peer fully qualified name (replication endpoint of active mode site)
site-name
Site name, used in k8s resource names
test-and-monitor-enabled          false
Test and Monitor enabled

Endpoints
=====
Name                               Value
Description                         -----
-----
api-manager-ui                    api-manager-ui.testsrv0231.subnet1.example.com
FQDN of API manager UI endpoint
cloud-admin-ui                    cloud-admin-ui.testsrv0231.subnet1.example.com
FQDN of Cloud admin endpoint
consumer-api                      consumer-api.testsrv0231.subnet1.example.com
FQDN of consumer API endpoint
hub
FQDN of Test and Monitor hub endpoint, only required if Test and Monitor is enabled
management-replication
```

```
FQDN of Management replication endpoint, only required if HA is enabled
platform-api          platform-api.testsrv0231.subnet1.example.com
FQDN of platform API endpoint
turnstile
FQDN of Test and Monitor turnstile endpoint, only required if Test and Monitor is enabled
```

4. Specify your deployment profile, the available profiles are described here: [Planning your deployment topology and profiles](#).

```
apicup subsys set mgmt deployment-profile=nlxc4.m16
```

Note: The deployment profiles that are shown in the **Description** column of the `apicup get` output are not correct. The available profiles are documented in [Planning your deployment topology and profiles](#).

5. Specify the API Connect license information:
- Specify the license ID:

```
apicup licenses accept <License_ID>
```

The `<License_ID>` is specific to the API Connect Program Name you purchased. To view the supported license IDs, see [API Connect licenses](#).

- Specify the license type:

```
apicup subsys set mgmt
license-use=<license_type>
```

The `license_type` must be either `production` or `nonproduction`. If not specified, the default value is `nonproduction`.

6. Optional: Configure backups of the subsystem. Note that once you set up scheduled backups, you can also run backups on-demand. Follow the instructions in [Configure your Management subsystem backup](#).
 Important: It is highly recommend that you configure backups and also take additional steps to ensure that your configuration and data can be restored in the event of a disaster event. See [Disaster recovery of the management subsystem on VMware](#).

7. Optional: Configure your logging.

Logging can be configured at a later time, but you must enable it before installation to capture the log events from the installation.

- Complete the procedure at [Configuring remote logging for a VMware deployment](#).
- Enter the following command to create the log file:

```
apicup subsys set mgmt additional-cloud-init-file=config_file.yml
```

8. Enter the following commands to update the `apiconnect-up-v10.yml` with the information for your environment:

- Set your search domain. Multiple search domains should be separated by commas.

```
apicup subsys set mgmt search-domain=your_search_domain
```

Where `your_search_domain` is the domain of your servers, entered in all lowercase. Setting this value ensures that your searches also append these values, which are based on your company's DNS resolution, at the end of the search value. A sample search domain is `mycompany.example.com`.

Ensure that the value for `your_search_domain` is resolved in the system's `/etc/resolv.conf` file to avoid "502" errors when accessing the Cloud Manager web site. For example:

```
# Generated by resolvconf
search your_search_domain ibm.com other.domain.com
```

- Set your domain name servers (DNS).

Supply the IP addresses of the DNS servers for your network. Use a comma to separate multiple server addresses.

```
apicup subsys set mgmt dns-servers=ip_address_of_dns_server[,ip_address_of_another_dns_server_if_necessary]
```

- Use `apicup` to set your endpoints.

You can use wildcard aliases or host aliases with your endpoints. Optionally, you can specify all endpoints with one `apicup` command.

Note: You cannot specify the underscore character "_" in domain names that are used in endpoints. See [API Connect configuration on VMware](#).

Table 1. Management subsystem endpoints

Setting	Endpoint host description
platform-api	Platform API endpoint. The host where your platform API calls are routed. <pre>apicup subsys set mgmt platform-api=platform-api.hostname.domain</pre>
consumer-api	Consumer API endpoint. The host where your consumer API calls are routed. <pre>apicup subsys set mgmt consumer-api=consumer-api.hostname.domain</pre>
cloud-admin-ui	Cloud admin user interface API endpoint. The host where your cloud administrator user-interface API calls are routed. <pre>apicup subsys set mgmt cloud-admin-ui=cloud-admin-ui.hostname.domain</pre>
api-manager-ui	API Manager user interface endpoint. The host where your API Manager API calls are routed. <pre>apicup subsys set mgmt api-manager-ui=api-manager-ui.hostname.domain</pre>
hub	The core microservice components for the Automated API behavior testing capability. This is an optional component and can be left unset if not used. <pre>apicup subsys set mgmt hub hub.hostname.domain</pre> Note: Automated API behavior testing is not supported in two data center disaster recovery configurations.
turnstile	Allows for communication between the Test pods and the API Management subsystem. This is part of the Automated API behavior testing capability. This is an optional component and can be left unset if not used. <pre>apicup subsys set mgmt turnstile turnstile.hostname.domain</pre> Note: Automated API behavior testing is not supported in two data center disaster recovery configurations.

- Set a Public key.

```
apicup subsys set mgmt ssh-keyfiles=path_to_public_ssh_keyfile
```

Setting this key enables you to use **ssh** with this key to log in to the virtual machine to check the status of the installation. You will perform this check in later in [Deploying the Management subsystem OVA file](#).

10. Set the password that you enter to log into your Management appliance for the first time.
 - a. **Important:** Review the requirements for creating and using a hashed password. See [Setting and using a hashed default password](#).
 - b. If you do not have a password hashing utility, install one.

Operating system	Command
Ubuntu, Debian, OSX	If the mkpasswd command utility is not available, download and install it. (You can also use a different password hashing utility.) On OSX, use the command: gem install mkpasswd .
Windows, Red Hat	If necessary, a password hashing utility for the Windows operating system, like OpenSSL

- c. Create a hashed password

Operating system	Command
Ubuntu, Debian, OSX	mkpasswd --method=sha-512 --rounds=4096 password
Windows, Red Hat	For example, using OpenSSL: openssl passwd -1 password . Note that you might need to add your password hashing utility to your path; for example, on Windows: set PATH=c:\cygwin64\bin;%PATH%

- d. Set the hashed password for your subsystem:

```
apicup subsys set mgmt default-password='hashed_password'
```

11. Optional: If the default IP ranges for the API Connect Kubernetes pod and the service networks conflict with IP addresses that must be used by other processes in your deployment, modify the API Connect values.

You can change the IP ranges of the Kubernetes pod and the service networks from the default values of 172.16.0.0/16 and 172.17.0.0/16, respectively. In the case that a /16 subnet overlaps with existing IPs on the network, a Classless Inter-Domain Routing (CIDR) as small as /22 is acceptable. You can modify these ranges during initial installation and configuration only. You cannot modify them once an appliance has been deployed. See [API Connect configuration on VMware](#).

- a. Update the IP range for the Kubernetes pod

```
apicup subsys set mgmt k8s-pod-network='new_pod_range'
```

Where *new_pod_range* is the new value for the range.

- b. Update the IP range for Service networks.

```
apicup subsys set mgmt k8s-service-network='new_service_range'
```

Where *new_service_range* is the new value for the range.

12. Add your hosts.

```
apicup hosts create mgmt hostname.domainname hd_password
```

Where the following are true:

- *hostname.domainname* is the fully qualified name of the server where you are hosting your Management service, including the domain information.
- *hd_password* is the password that the Linux Unified Key Setup uses to encrypt the storage for your Management service. This password is hashed when it is stored on the server or in the ISO. Note that the password is base64 encoded when stored in **apiconnect-up-v10.yml**.

Repeat this command for each host that you want to add.

Note: Host names cannot be changed on a cluster after the initial installation.

13. Create your interfaces.

```
apicup iface create mgmt hostname.domainname physical_network_id host_ip_address/subnet_mask network_gateway_ip_address
```

Where *physical_network_id* is the network interface ID of your physical server. The value is most often **eth0**. The value can also be **ethx**, where **x** is a number identifier.

The format is similar to this example: **apicup iface create mgmt myHostname.domain eth0 192.0.2.10/255.255.255.0 192.0.2.1**.

Note: The *network_gateway_ip_address* is the network gateway (not a DataPower Gateway). If you are creating multiple network interfaces, each one must be on a different subnet with a different gateway.

14. Optional: Use **apicup** to view the configured hosts:

```
apicup hosts list mgmt
testsrv0231.subnet1.example.com
Device IP/Mask Gateway
eth0 1.2.152.231/255.255.254.0 1.2.152.1
```

Note: This command might return the following messages, which you can ignore:

```
* host is missing traffic interface
* host is missing public interface
```

Note: If you are configuring two data center deployment, continue with the installation instructions in [Installing a two data center deployment on VMware](#).

15. Optional: Verify that the configuration settings are valid.

```
apicup subsys get mgmt --validate
```

The output lists each setting and adds a check mark after the value once the value is validated. If the setting lacks a check mark and indicates an invalid value, reconfigure the setting. See the following sample output.

Appliance settings

=====

```
Name          Value
----          -
additional-cloud-init-file
✓
data-device      sdb
✓
default-password
$6$rounds=4096$vtcqpAVK$dzqrOeYP33WTVtug38Q4R1d518TmdQgezZtnkX/PFwkzTiZ2S0CqNRr1S4b08tOc4p.OEg4BtzBe/r8RAk.gw/ ✓
dns-servers      [8.8.8.8]
✓
extra-values-file
✓
k8s-pod-network  172.16.0.0/16
✓
k8s-service-network  172.17.0.0/16
✓
public-iface     eth0
✓
search-domain    [subnet1.example.com]
✓
ssh-keyfiles     [/home/vsphere/.ssh/id_rsa.pub]
✓
traffic-iface    eth0
✓
```

Subsystem settings

=====

```
Name          Value
----          -
deployment-profile  nlxc4.m16
✓
license-use        production
✓
multi-site-ha-enabled  false
✓
multi-site-ha-mode   active
✓
replication-peer-fqdn
✓
site-name
✓
test-and-monitor-enabled  false
✓
```

Endpoints

=====

```
Name          Value
----          -
api-manager-ui  api-manager-ui.testsrv0231.subnet1.example.com
✓
cloud-admin-ui  cloud-admin-ui.testsrv0231.subnet1.example.com
✓
consumer-api    consumer-api.testsrv0231.subnet1.example.com
✓
hub
✓
management-replication
✓
platform-api    platform-api.testsrv0231.subnet1.example.com
✓
turnstile
✓
```

16. Create your ISO file.

```
apicup subsys install mgmt --out mgmtplan-out
```

The `--out` parameter and value are required.

In this example, the ISO file is created in the `myProject/mgmtplan-out` directory.

If the system cannot find the path to your software that creates ISO files, create a path setting to that software by running a command similar to the following command:

Operating system	Command
Ubuntu, Debian, OSX	<code>export PATH=\$PATH:/Users/your_path/</code>
Windows, Red Hat	<code>set PATH="c:\Program Files (x86)\cdrtools";%PATH%</code>

17. Deploy the subsystem ISO. Continue with [Deploying the Management subsystem OVA file](#).

Deploying the Management subsystem OVA file

After you have specified configuration settings and created an ISO, deploy the subsystem OVA into an appliance VM.

Before you begin

You must have completed the configuration of the project file entries for the management subsystem, in [Configuring the Management subsystem](#).

About this task

Procedure

1. Log into the VMWare vSphere Web Client.
 2. Using the VSphere Navigator, navigate to the directory where you are deploying the OVA file.
 3. Right-click the directory and select Deploy OVF Template.
 4. Complete the Deploy OVF Template wizard.
 - a. Select the Management subsystem .OVA file template by navigating to the location where you downloaded the file in [First steps for deploying in a VMware environment](#).
 - b. Enter a name and location for your file.
 - c. Select a resource for your template.
 - d. Review the details for your template.
 - e. Select the size of your configuration.
 - f. Select the storage settings.

Important: Select at least 200 GB for the data disk size. Note that the boot disk needs 100 GB, so you need a total of 300 GB for the API Connect management subsystem.
 - g. Select the networks.
 - h. Customize the Template, if necessary.
 - i. Review the details to ensure that they are correct.
 - j. Select Finish to deploy the virtual machine.
- Note: Do not change the OVA hardware version, even if the VMware UI shows a Compatibility range that includes other versions. See [Requirements for initial deployment on VMware](#).
- The template creation appears in your Recent Tasks list.
5. Select the Storage tab in the Navigator.
 6. Navigate to your datastore.
 7. Upload your ISO file.
 - a. Select the Navigate to the datastore file browser icon in the icon menu.
 - b. Select the Upload a file to the Datastore icon in the icon menu.
 - c. Navigate to the ISO file that you created in your project.

It is the myProject/mgmtplan-out
 - d. Upload the ISO file to the datastore.
 8. Leave the datastore by selecting the VMs and Templates icon in the Navigator.
 9. Locate and select your virtual machine.
 10. Select the Configure tab in the main window.
 11. Select Edit...
 - a. On the Virtual Hardware tab, select CD/DVD Drive 1.
 - b. For the Client Device, select Datastore ISO File.
 - c. Find and select your datastore in the Datastores category.
 - d. Find and select your ISO file in the Contents category.
 - e. Select OK to commit your selection and exit the Select File window.
 - f. Ensure that the Connect At Power On check box is selected.

Tip:

 - Expand the CD/DVD drive 1 entry to view the details and the complete Connect At Power On label.
 - Note that VMware related issues with ISO mounting at boot may occur if Connect At Power On
 - g. Select OK to commit your selection and close the window.
 12. Start the virtual machine by selecting the play button on the icon bar.

The installation might take several minutes to complete, depending on the availability of the system and the download speed.
 13. Verify the API Connect installation. Continue with [Verify installation of the Management subsystem](#)

Verify installation of the Management subsystem

Verify that deployment of the management subsystem OVA file succeeded.

Before you begin

You must have completed [Deploying the Management subsystem OVA file](#).

About this task

After you configure an ISO and deploy it with the management subsystem ova, as described in [Deploying the Management subsystem OVA file](#), verify that the subsystem installed correctly and is functional.

Procedure

1. Log in to the virtual machine by using an SSH tool to check the status of the installation:
 - a. Enter the following command to connect to *mgmt* using SSH:

```
ssh ip_address -l apicadm
```

- You are logging in with the default ID of *apicadm*, which is the API Connect ID that has administrator privileges.
- b. Select Yes to continue connecting.
Your host names are automatically added to your list of hosts.
 - c. Run the **apic status** command to verify that the installation completed and the system is running correctly.
Note that after installation completes, it can take several minutes for all servers to start. If you see the error message **Subsystems not running**, wait a few minutes, try the command again, and review the output in the **Status** column.

The command output for a correctly running Management system is similar to the following lines:

```
apicadm@testsys0181:~$ sudo apic status

INFO[0001] Log level: info
Cluster members:
- testsys0164.subnet1.example.com (1.1.1.1)
  Type: BOOTSTRAP MASTER
  Install stage: DONE
  Upgrade stage: NONE
  Docker status:
    Systemd unit: running
  Kubernetes status:
    Systemd unit: running
    Kubelet version: testsys0164 (4.4.0-137-generic) [Kubelet v1.10.6, Proxy v1.10.6]
    Etd status: pod etcd-testsys0164 in namespace kube-system has status Running
    Addons: calico, dns, helm, kube-proxy, metrics-server, nginx-ingress,
- testsys0165.subnet1.example.com (1.1.1.2)
  Type: MASTER
  Install stage: DONE
  Upgrade stage: NONE
  Docker status:
    Systemd unit: running
  Kubernetes status:
    Systemd unit: running
    Kubelet version: testsys0165 (4.4.0-137-generic) [Kubelet v1.10.6, Proxy v1.10.6]
    Etd status: pod etcd-testsys0165 in namespace kube-system has status Running
    Addons: calico, kube-proxy, nginx-ingress,
- testsys0181.subnet1.example.com (1.1.1.3)
  Type: MASTER
  Install stage: DONE
  Upgrade stage: NONE
  Docker status:
    Systemd unit: running
  Kubernetes status:
    Systemd unit: running
    Kubelet version: testsys0181 (4.4.0-137-generic) [Kubelet v1.10.6, Proxy v1.10.6]
    Etd status: pod etcd-testsys0181 in namespace kube-system has status Running
    Addons: calico, kube-proxy, nginx-ingress,
Etd cluster state:
- etcd member name: testsys0164.subnet1.example.com, member id: 11019072309842691371,
  cluster id: 5154498743703662183, leader id: 11019072309842691371, revision: 21848, version: 3.1.17
- etcd member name: testsys0165.subnet1.example.com, member id: 541472388445093633,
  cluster id: 5154498743703662183, leader id: 11019072309842691371, revision: 21848, version: 3.1.17
- etcd member name: testsys0181.subnet1.example.com, member id: 3261849123413063575,
  cluster id: 5154498743703662183, leader id: 11019072309842691371, revision: 21848, version: 3.1.17

Pods Summary:

NODE          NAMESPACE   NAME                                     READY
STATUS        REASON
testsys0165   kube-system  calico-node-jp8zv                       2/2
Running
testsys0164   kube-system  calico-node-pjjgh                       2/2
Running
testsys0181   kube-system  calico-node-ssb9w                       2/2
Running
testsys0164   kube-system  coredns-87cb95869-9nvdr                 1/1
Running
testsys0164   kube-system  coredns-87cb95869-r9q8w                 1/1
Running
testsys0164   kube-system  etcd-testsys0164                        1/1
Running
testsys0165   kube-system  etcd-testsys0165                        1/1
Running
testsys0181   kube-system  etcd-testsys0181                        1/1
Running
testsys0165   kube-system  ingress-nginx-ingress-controller-92mkz  1/1
Running
testsys0181   kube-system  ingress-nginx-ingress-controller-kt9sr  1/1
Running
testsys0164   kube-system  ingress-nginx-ingress-controller-p7x55  1/1
Running
testsys0164   kube-system  ingress-nginx-ingress-default-backend-6f58fb5f56-t27gx 1/1
Running
testsys0164   kube-system  kube-apiserver-testsys0164              1/1
Running
testsys0165   kube-system  kube-apiserver-testsys0165              1/1
Running
testsys0181   kube-system  kube-apiserver-testsys0181              1/1
Running
testsys0164   kube-system  kube-apiserver-proxy-testsys0164        1/1
Running
testsys0165   kube-system  kube-apiserver-proxy-testsys0165        1/1
Running
testsys0181   kube-system  kube-apiserver-proxy-testsys0181        1/1
Running
```

testsys0164	kube-system	kube-controller-manager-testsys0164	1/1
Running			
testsys0165	kube-system	kube-controller-manager-testsys0165	1/1
Running			
testsys0181	kube-system	kube-controller-manager-testsys0181	1/1
Running			
testsys0165	kube-system	kube-proxy-7gqpw	1/1
Running			
testsys0181	kube-system	kube-proxy-8hc8t	1/1
Running			
testsys0164	kube-system	kube-proxy-bhgqc	1/1
Running			
testsys0164	kube-system	kube-scheduler-testsys0164	1/1
Running			
testsys0165	kube-system	kube-scheduler-testsys0165	1/1
Running			
testsys0181	kube-system	kube-scheduler-testsys0181	1/1
Running			
testsys0164	kube-system	metrics-server-6fbfb84cdd-1ffxc	1/1
Running			
testsys0164	kube-system	tiller-deploy-84f4c8bb78-xxfds	1/1
Running			

2. Ensure that you have saved all the necessary artifacts to enable you to restore the management subsystem in case of a disaster event. See [Preparing the management subsystem for disaster recovery on VMware](#).
3. Verify you can access the API Connect Cloud Manager. Enter the URL in your browser. The syntax is `https://<hostname.domain>/admin`. For example:

```
https://cloud-admin-ui.testsrv0231.subnet1.example.com/admin
```

The first time that you access the Cloud Manager user interface, you enter `admin` for the user name and `7iron-hide` for the password. You will be prompted to change the Cloud Administrator password and email address. See [Accessing the Cloud Manager user interface](#).

4. If you configured an S3 backup during installation, verify that the configuration succeeded. See [Verify configuration for s3 backup](#).

Deploying the Analytics virtual appliance

Install the IBM® API Connect Analytics subsystem.

Before you begin

Review the section on [Planning your Analytics deployment](#) to ensure that you understand the choices for configuring the topology, disk space, and other elements of the Analytics subsystem. It is important to follow these steps in sequence, finishing with *Deploying the Analytics subsystem*.

- [Creating the analytics-extra-values file](#)
Create the installation extra-values file for API Connect Analytics so that you can configure additional features.
- [Enabling additional analytics features](#)
Update the `analytics-extra-values.yaml` file to enable and configure additional features of your API Connect Analytics deployment.
- [Deploying the Analytics subsystem](#)
Define your analytics subsystem, create an ISO file for each analytics node, and deploy the analytics nodes using the Analytics OVA file on a VMware virtual server.

Creating the analytics-extra-values file

Create the installation extra-values file for API Connect Analytics so that you can configure additional features.

Before you begin

Create the `analytics-extra-values.yaml` file if you plan to use features such as persistent queue, data offloading to third-party systems, modifying incoming data, or disabling internal storage. For information on the available configuration options, see [Planning your Analytics deployment](#). This file can be created and features configured or reconfigured after installation, except for disabling internal storage, which must be set before installation and cannot be changed afterward.

About this task

To apply configuration options to the Analytics subsystem during installation, create an extra-values file where you can specify your settings. Use the `analytics-extra-values.yaml` file to apply configuration settings when you install, upgrade, and update the configuration of your Analytics subsystem.

Procedure

1. Create a file called `analytics-extra-values.yaml` and open it for editing.
2. For the first line in the file, add:

```
spec:
```

All of your configuration options must follow this line, and must be indented with 2 spaces (or more for nested values).

3. Save the file.

What to do next

Add configuration options to the `analytics-extra-values.yaml` file as explained in [Enabling additional analytics features](#). If you do not want to configure additional analytics features, continue to [Deploying the Analytics subsystem](#).

- [Reference: Fields in analytics-extra-values](#)
Use the table of `analytics-extra-values.yaml` file settings to configure the analytics features you require.

Reference: Fields in analytics-extra-values

Use the table of `analytics-extra-values.yaml` file settings to configure the analytics features you require.

Table 1. Fields that you can set in the `analytics-extra-values` file

Field	Type	Description	Comments
<code>external: offload: filter</code>	String	Filter configuration for external offload.	See Offloading data to a third-party system .
<code>external: offload: enabled</code>	Boolean	Enables analytics offload feature.	See Offloading data to a third-party system .
<code>external: offload: output</code>	String	Output configuration for external offload.	See Offloading data to a third-party system .
<code>external: offload: passwordSecretName</code>	String	The name of the secret that has the password to access the certificate keystore.	See Offloading data to a third-party system .
<code>external: offload: secretName</code>	String	The name of the secret that contains the certificates used for connecting to the offload target.	See Offloading data to a third-party system .
<code>ingestion: filter</code>	String	Filter to modify API event data that is stored internally (not offloaded).	See Modifying your analytics data .
<code>ingestion: queue</code>	Object	Persistent queue settings. Must be set to <code>persisted</code> if present.	See Analytics persistent queue .
<code>ingestion: queue: volumeClaimTemplate: volumeSize</code>	String	Storage size for your persistent queue.	See Analytics persistent queue .
<code>ingestion: queue: volumeClaimTemplate: storageClassName</code>	String	Storage class name for your persistent storage. Must be set to <code>local-storage</code> .	See Analytics persistent queue .
<code>ingestion: pipelineWorkers</code>	4 <= integer <= 128	Number of pipeline workers.	More workers mean more CPU and memory requirements, but gives you the benefit of extra threads to pull data off the ingestion http interface and process it for storage. This has diminishing returns as the size increases, and should usually be multiple of the number of cores on your hardware. Adjust with caution.
<code>ingestion: pipelineBatchSize</code>	10 <= integer <= 10000	Batch size of a pipeline worker.	Pulling more or less data off the http interface per worker can have benefits. Recommended values are 125, 250, and 500. Increasing the value requires more memory.
<code>ingestion: pipelineBatchDelay</code>	10 <= integer <= 5000	Batch delay of a pipeline worker.	How long to wait for the batch size to fill before leaving the http interface to process for storage. Having threads wait too long causes idle threads in your system; however, setting this value too low decreases performance. Adjust with caution.
<code>ingestion: configReloadAutomatic</code>	Boolean	Whether to automatically reload the pipeline on configuration changes.	Typically, when the underlying files for the pipeline are modified, the pipeline process automatically reloads and picks up the changes. However, in Kubernetes environments where the pipeline is mapped to configmaps, this is unreliable. Ingestion microservice restarts are usually required.
<code>ingestion: pipelineUnsafeShutdown</code>	Boolean	Whether to allow unsafe shutdown of the pipeline.	Typically, the pipeline attempts to empty all workers' batches before shutdown to prevent data loss. You might need to enable this setting if the pipeline is badly configured and refuses to respond to shut down requests (uncommon). Adjust with caution.
<code>storage: enabled</code>	Boolean	Set false to disable internal storage.	See Disabling internal storage .

Enabling additional analytics features

Update the `analytics-extra-values.yaml` file to enable and configure additional features of your API Connect Analytics deployment.

About this task

Based on the planning decisions you made [Planning your Analytics deployment](#), configure your deployment options by completing the steps in the following topics to enable the persistent queue, configure data offloading, disable internal storage, and add ingestion filters.

- [Analytics persistent queue](#)
For enhanced reliability in your analytics data pipeline, enable the persistent queue feature.
- [Offloading data to a third-party system](#)
Update the `analytics-extra-values.yaml` file to configure data offloading from your API Connect Analytics deployment to a third-party system.
- [Disabling internal storage](#)
Update the `analytics-extra-values.yaml` file to offload all data and disable local storage in your API Connect analytics deployment.
- [Modifying your analytics data](#)
Update the `analytics-extra-values.yaml` file to customize data in your API Connect Analytics deployment.

Analytics persistent queue

For enhanced reliability in your analytics data pipeline, enable the persistent queue feature.

About this task

The Logstash persistent queue is an optional feature that provides enhanced reliability in the data pipeline. With the persistent queue enabled, the ingestion pod buffers incoming API event data to persistent storage so that no events are lost if the downstream API Connect services become unavailable.

For example, if the analytics storage service is restarted, any incoming API events are kept in persistent storage and sent on to the storage service when it is available again. If the persistent queue is not enabled, then the ingestion pod rejects further API event data coming from the gateway.

Another benefit of this feature is better handling of unexpected traffic spikes. The only disadvantages are higher CPU and persistent storage requirements.

Procedure

1. Open the `analytics-extra-values.yaml` file for editing.
Remember: All additions to the file must appear after the `spec:` statement.
2. Add the following section to the file:

```
ingestion:
  queue:
    type: persisted
    volumeClaimTemplate:
      storageClassName: local-storage
      volumeSize: <storage size>
```

Where:

- `<storage size>` is the maximum size for your persistent queue. For example, `30Gi`.

3. Save the file.

What to do next

If you do not want to add any more configuration options, skip to the topic [Deploying the Analytics subsystem](#).

Otherwise, continue adding configuration options to the file as explained in the following topics.

Offloading data to a third-party system

Update the `analytics-extra-values.yaml` file to configure data offloading from your API Connect Analytics deployment to a third-party system.

About this task

For information about offloading data from your Analytics deployment, see [Planning to offload data to a third-party system](#).

Procedure

1. Open the `analytics-extra-values.yaml` file for editing.
Remember: All additions to the file must appear after the `spec:` statement.
2. Add the following section to the file:

```
spec:
  external:
    offload:
      enabled: true
      output: |
        $EXTERNAL_OFFLOAD_OUTPUT
```

where `$EXTERNAL_OFFLOAD_OUTPUT` is the plugin output for the target third-party system.

For example, the following example enables offloading to HTTP:

```
spec:
  external:
    offload:
      enabled: true
      output: |
        http {
          url => "example.com"
          http_method => "post"
          codec => "json"
          content_type => "application/json"
          headers => {
            "x-acme-index" => "8"
            "x-acme-key" => "0d5d259f-b8c5-4398-9e58-77b05be67037"
          }
        }
      id => "offload_http"
    }
```

For examples of output plugins, see [Sample output plugins for offloading data](#).

- Optional: Modify the output plugin to include certificates for connecting to a secure endpoint.

Important: Certificates can only be applied after the Analytics subsystem is deployed. If you want to configure an output plugin that references a certificate, you must install the Analytics subsystem first, then create a secret and apply it to the Analytics deployment, configure the output plugin, and re-install Analytics. For instructions on the correct procedure for applying a certificate to the Analytics deployment, see [Providing a custom certificate for Analytics offload](#). If your third-party endpoint requires a particular certificate, you can configure the certificate and then reference it in the offload plugin. Include the `secretName` as well as the `cacert`, `client_cert`, and `client_key` settings.

For example, the following output plugin enables offloading to HTTP and provides information on the certificate that is needed for authenticating with the endpoint:

```
spec:
  external:
    offload:
      secretName: offload-certificates
      enabled: true
      output: |
        http {
          url => "https://example.com:443"
          http_method => "post"
          codec => "json"
          content_type => "application/json"
          headers => {
            "x-acme-index" => "8"
            "x-acme-key" => "0d5d259f-b8c5-4398-9e58-77b05be67037"
          }
        }
      id => "offload_http"
      cacert => "/etc/velox/external_certs/offload/cacert.pem"
      client_cert => "/etc/velox/external_certs/offload/client.pem"
      client_key => "/etc/velox/external_certs/offload/client.key"
    }
```

The secret and its defined files are mounted to the `/etc/velox/external_certs/offload` directory.

- Optional: Add a `filter` after the output plugin to modify data prior to offloading it. The following example filter removes the `query_string` field:

```
spec:
  external:
    offload:
      enabled: true
      output: |
        http {
          url => "example.com"
          http_method => "post"
          codec => "json"
          content_type => "application/json"
          headers => {
            "x-acme-index" => "8"
            "x-acme-key" => "0d5d259f-b8c5-4398-9e58-77b05be67037"
          }
        }
      id => "offload_http"
    }
  filter: |
    ruby {
      remove_field => ["query_string"]
    }
  }
```

For information about modifying offloaded data, see [Planning to modify your analytics data](#).

- Save the file.

What to do next

If you do not want to add any more configuration options, skip to the topic on [Deploying the Analytics subsystem](#).

Otherwise, continue adding configuration options to the file as explained in the following topics.

Attention: The third-party endpoint must be available at all times to prevent the loss of analytics data.

- [Sample output plugins for offloading data](#)
Review sample output plugins to see how to code your own plugins for offloading data from API Connect Analytics.
- [Providing a custom certificate for Analytics offload](#)
Prepare a private or self-signed certificate for offloading data.

Sample output plugins for offloading data

Review sample output plugins to see how to code your own plugins for offloading data from API Connect Analytics.

Configure output plugins that direct analytics data to your third-party systems. An output plugin specifies where analytics data is offloaded, plus any information required to access that system, such as certificate information. The output plugin for displaying analytics data in the Analytics user interface is configured by default.

Referencing a certificate in an output plugin

If your third-party system requires a private or self-signed certificate, you can include a reference to the certificate in your output plugin. See the second HTTP output plugin for an example showing how to reference the certificate.

Important: Analytics must be deployed before you attempt to use the certificate. If you want to configure an output plugin that references a certificate, you must install the Analytics subsystem first, then create a secret and apply it to the Analytics deployment, configure the output plugin, and re-install Analytics. For instructions on the correct procedure for applying a certificate to the Analytics deployment, see [Providing a custom certificate for Analytics offload](#).

HTTP

To configure data offload to an HTTP server, you must provide the server URL. You can optionally add standardized or custom HTTP headers to define additional operating parameters.

The following example configures an HTTP output plugin.

```
http {
  url => "example.com"
  http_method => "post"
  codec => "json"
  content_type => "application/json"
  headers => {
    "x-acme-index" => "8"
    "x-acme-key" => "0d5d259f-b8c5-4398-9e58-77b05be67037"
  }
  id => "offload_http"
}
```

The following example configures an HTTP output plugin that references a custom certificate for accessing the third-party endpoint.

```
http {
  url => "https://example.com:443"
  http_method => "post"
  codec => "json"
  content_type => "application/json"
  headers => {
    "x-acme-index" => "8"
    "x-acme-key" => "0d5d259f-b8c5-4398-9e58-77b05be67037"
  }
  id => "offload_http"
  cacert => "/etc/velox/external_certs/offload/cacert.pem"
  client_cert => "/etc/velox/external_certs/offload/client.pem"
  client_key => "/etc/velox/external_certs/offload/client.key"
}
```

For more information on offloading to HTTP, refer to the Logstash documentation on configuring the [HTTP output plugin](#).

Elasticsearch

To configure data offload to an Elasticsearch cluster, you must provide one or more server URLs, define how indices should be created by specifying a dynamically configured name, specify the number of primary shards for storing the indexed documents, and specify a number of replica shards for redundancy. You can optionally specify server authentication credentials.

The naming syntax that you specify for your indices determines how many indices are created and how they are characterized. For example, you can define indices that are created based on a date pattern (for example, daily or weekly), or indices that are created to store data by provider organization or by API name.

The following example configures an Elasticsearch output plugin.

```
elasticsearch {
  hosts => "https://example.com:443"
  index => "apiconnect"
  ssl => true
  cacert => "/etc/pki/tls/cert.pem"
}
```

For more information on offloading to Elasticsearch, refer to the Elasticsearch documentation on the [Elasticsearch output plugin](#).

Apache Kafka

To configure data offload to a Kafka cluster, you must provide host and port connection details for one or more servers, and the name of the Kafka topic to which you want to publish the offloaded data. For logging and monitoring purposes within Kafka, you can optionally specify a string identifier by which API Connect can be uniquely identified.

The following example configures a Kafka output plugin.

```
kafka {
  topic_id => "test"
  bootstrap_servers => "example.com:9093"
  codec => "json"
  id => "kafka_offload"
  ssl_truststore_location => "/usr/lib/jvm/jre/lib/security/cacerts"
  ssl_truststore_password => "changeit"
  ssl_truststore_type => "JKS"
  security_protocol => SSL
}
```

For more information on offloading to Apache Kafka, refer to the Logstash documentation on the [Kafka output plugin](#).

Syslog

To configure data offload to Syslog, you must provide host and port connection details for the server.

Restriction: Syslog does not support the use of chained client certificates for TLS profiles.

The following example configures a Syslog output plugin.

```
syslog {
  host => "example.com"
  port => 601
  protocol => "ssl-tcp"
  id => "offload_syslog"
  appname => "apiconnect"
  msgid => "%{org_id}"
  facility => "log_audit"
  severity => "informational"
  codec => "json"
  ssl_cacert => "/etc/pki/tls/cert.pem"
}
```

Attention: Due to a third-party issue with Logstash, most of the facility values are not set correctly on the offload server. As a temporary workaround, set the facility to "user-level" to ensure that the corresponding value is correctly sent to the offload server. When Logstash corrects this problem, you must update the offload configuration to change the value to a more appropriate setting.

For more information on offloading to Syslog, refer to the Logstash documentation on the [Syslog output plugin](#).

OpenSearch

To configure data offload to OpenSearch: <https://opensearch.org/docs/latest/clients/logstash/ship-to-opensearch/>

Providing a custom certificate for Analytics offload

Prepare a private or self-signed certificate for offloading data.

Before you begin

Install the Analytics subsystem on VMware as explained in [Deploying the Analytics subsystem](#). Analytics must be deployed before you attempt to use the certificate.

When you re-install Analytics after applying the certificate, you can upload the analytics-extra-values.yaml file to include additional configuration options.

You must have the certificate in either of these formats:

- Base64 encoded **pem** file format.
- JKS file.

Procedure

1. Login to your analytics VM.
 - a. SSH to your analytics VM, or one of the VMs if you have a three replica deployment.

```
ssh apicadm@<analytics VM FQDN>
```

- b. Switch to root user:

```
sudo -i
```

- c. If you are using a JKS file, create a file to contain your certificate.

```
echo "<certificate>" > <certificate filename>
```

2. If you are using a JKS file, encode the contents in base64.

```
cat <certificate filename> | base64
```

Copy the output so that you can paste it into the secret in the next step.

3. Create a Kubernetes secret to contain the certificate.

a. Create a file called `analytics_certs.yaml`.

The method for defining the certificate in the secret depends on whether you are using a PEM file or JKS file.

- JKS file: In the `data` section of the secret, paste the encoded output from step 2 as shown:

```
apiVersion: v1
kind: Secret
metadata:
  name: offload-certificates
data:
  keystore.jks: "<encoded JKS string>"
```

- PEM file: In the `stringData` section of the secret, type a name for the key or certificate (you can choose the name), and paste the contents of the PEM file.

```
apiVersion: v1
kind: Secret
metadata:
  name: offload-certificates
stringData:
  cacert.pem: |-
-----BEGIN CERTIFICATE-----
MIIDIDCCAomgAwIBAgIENd70zzANBgkqhkiG9w0BAQUFADBOMQswCQYDVQQGEwJVUzEQMA4GA1UE
ChMHRXFlaWZheDEtMCsGA1UECzMkRXFlaWZheCBTZWN1cmUgQ2VydG1maWNhdGUgQXV0aG9yaXR5
MB4XDk4MDgYmJmE2NDE1MVoXDTQ4MDgYmJmE2NDE1MVoXDTQ4MDgYmJmE2NDE1MVoXDTQ4MDgY
B0VxdWlmYXgXLTArBgNVBAsTJEVxdWlmYXgGU2VjdXJlIEN1cnRpZml1eXJ1IEFlidGhvcml0eTCB
nzANBgkqhkiG9w0BAQEFAAOBjQAwGyKCGYEAwV2xWgcIYu6gmi0fCG2RFGiYCh7+2gRvE4RiIcPR
fM6fBeC4AfBONOziipUEZKzxa1NfBbPLZ4C/QgKO/t0BCezhABRP/PvwDN1Dulsr4R+AcJkVV5MW
8Q+XarfCaCMczE1ZMKxRHjuvK9buY0V7xd1fUNLjUA86iOe/FP3gx7kCAwEAaAaOCAQkwggEFMHAG
A1UdHwRPMGcwZaBjoGGkXzBdMQswCQYDVQQGEwJVUzEQMA4GA1UEChMHRXFlaWZheDEtMCsGA1UE
CzMkRXFlaWZheCBTZWN1cmUgQ2VydG1maWNhdGUgQXV0aG9yaXR5MQ0wCwYDVQQDEwRDUkwMB0G
A1UdEAQTMGBGDzIwMTgwODIyMTY0MTUxWjALBgNVHQ8EBAMCAQYwHwYDVDR0jBBGwFoAUSOZo+SvS
spXXR9gjlBBPM5iQn9QwHQYDVR0OBBYEFjmaPkr0rKV10fYIyAQZtOYkKJ/UMAwGA1UdEwQFMAMB
Af8wGgYJKoZIhvcZ9BOEABA0CxsFVjMuMGMDAgBAMA0GCSqGSIb3DQEBBQUAA4GBAFjOKer89961
zgK5F7WF0bnj4JXMJTENAKaSbn+2kmOeUJXRmm/kEd5jhW6Y7qj/WsjTVbJmcVfewChrPSqnI0kB
BIZCe/zuf6IWUrvnZ9NA2zsmWLIodz2uFHdh1voqZiegDfqnclzqcPGUIWVEX/z87y1oqaKHee95
70+sB3c4
-----END CERTIFICATE-----
```

You can add multiple certificates, and can set the names to what you want.

Note: Kubernetes, OpenShift, and Cloud Pak for Integration users: For each key included in `analytics_certs.yaml`, a file based on the corresponding key name is created in the ingestion pod container at `/etc/velox/external_certs/offload`. The examples here would result in the files `keystore.jks` and `cacert.pem`.

b. Save the file.

4. Optional: Create a password secret.

If you have sensitive data such as a certificate password that must be available in text format, you can store it in another secret.

If your password is not sensitive (such as the JKS default password), then you can supply it directly in the Logstash configuration, and skip this step.

a. Create a file called `offload_env_var.yaml`.

b. In the `stringData` section, set the property `jks_password.txt` to your password (in plain text):

```
apiVersion: v1
kind: Secret
metadata:
  name: offload-jks-password
stringData:
  jks_password.txt: <password>
```

c. Run the following command:

```
kubectl apply -f offload_env_var.yaml -n <namespace>
```

VMware users can omit the `-n <namespace>`.

5. Apply the secret to the cluster where the analytics subsystem is installed.

Run the following command to update the cluster:

```
kubectl apply -f analytics_certs.yaml -n <namespace>
```

VMware users can omit the `-n <namespace>`.

6. Add the secret (and password secret if you created one) to the offload section in your `analytics-extra-values.yaml`

a. Edit the `analytics-extra-values.yaml` file, and add the secrets:

```
offload:
  enabled: true
  output: |
    kafka {
      ...
    }
  secretName: offload-certificates
  passwordSecretName: offload-jks-password
```

b. Ensure that your `extra-values-file` property is set to point to your `analytics-extra-values.yaml` file:

```
apicup subsys set <analytics subsystem> extra-values-file analytics-extra-values.yaml
```

c. Apply the `analytics-extra-values.yaml` to your analytics subsystem:

```
apicup subsys install <analytics subsystem>
```

Disabling internal storage

Update the `analytics-extra-values.yaml` file to offload all data and disable local storage in your API Connect analytics deployment.

About this task

For information about disabling internal storage in your Analytics deployment, see [Planning to disable internal storage](#).

Warning: It is not possible to enable internal storage after installation. If you install with internal storage disabled and then decide you want to enable it, you will need to reinstall the analytics subsystem.

Procedure

1. Required: Configure data offloading as explained in [Offloading data to a third-party system](#). The analytics subsystem will not start if storage is disabled and offloading not configured.
2. Open the `analytics-extra-values.yaml` file for editing.
3. Add the `storage` section as shown in the following example:

```
storage:
  enabled: false
```

4. Save the file.

What to do next

If you do not want to add any more configuration options, skip to the topic on [Deploying the Analytics subsystem](#).

Otherwise, continue adding configuration options to the file as explained in the following topics.

Modifying your analytics data

Update the `analytics-extra-values.yaml` file to customize data in your API Connect Analytics deployment.

Before you begin

Review the use of filters for modifying analytics data, and make sure that you understand the restrictions. For information about customizing data, see [Planning to modify your incoming analytics data](#).

About this task

You can define filters that add, modify, or remove fields from incoming API event records.

Procedure

1. Open the `analytics-extra-values.yaml` file for editing.
Remember: All additions to the file must appear after the `spec:` statement.
2. Add a `filter` definition.
Filters for API events that are offloaded to an external system are defined in the `external.offload` section. Filters for API events that are stored in the API Connect OpenSearch database are defined in the `ingestion` section. See the following examples:

Ingestion filters - for API event data stored by API Connect
Add the filter definition to the `ingestion` section:

```
spec:
  ingestion:
    ...
    filter: |
      mutate {
        remove_field => ["query_string"]
      }
```

Offload filters - for API event data that is sent to a third-party system.
Add the filter definition to the `external.offload` section:

```
spec:
  external:
    offload:
      filter: |
        mutate {
          remove_field => ["query_string"]
        }
```

For more example filters, see [Sample filters for modifying analytics data](#).

3. Optional: Include geoIP fields in your analytics data, for both stored and offloaded data, by inserting `geoIPEnabled: true` in the ingestion section:

```
ingestion:
  ...
  geoIPEnabled: true
```

By default the `client_geoip` and `gateway_geoip` fields are not included in the analytics data. You do not need to add `geoIPEnabled` to the `external.offload` section. Enabling `geoIP` in the `ingestion` section enables it for both data routes.

Note: To ensure the value in the `client_geoip` field is accurate, the gateway must receive the `X-Forwarded-For` header in all API calls. Check with the administrator of your deployment environment to ensure that the `X-Forwarded-For` header is passed to your gateways. For example, in a Kubernetes environment where NINGX ingresses are used, configure the NGINX ingress that is used by your gateway to use the `X-Forwarded-For` header: [NGINX Configuration "use-forwarded-headers"](#).

4. Save the file.

What to do next

If there are no more features you want to configure in the extra-values file, proceed to deployment of the OVA: [Deploying the Analytics subsystem](#).

- [Sample filters for modifying analytics data](#)
Review sample filters to see how to code your own filters for refining API Connect Analytics data by adding fields, removing fields, or modifying field contents.

Sample filters for modifying analytics data

Review sample filters to see how to code your own filters for refining API Connect Analytics data by adding fields, removing fields, or modifying field contents.

You can use these sample filters in the `ingestion.filter` and the `external.offload.filter` configurations.

Adding a new field

You can add custom data to API event data by adding a field to the logging document. For example, if you want to search or visualize data that is stored in request and response headers, you can add fields to a filter to include that information.

To avoid naming conflicts with future or current analytics fields, include a prefix that ensures your new field name is unique. For example, instead of naming the field `employee_num` you might name it `x_mycompany_employee_num`.

The following example copies the contents of the `X-Employee-Num` field from the request header and adds it to the `x_mycompany_employee_num` field in the event data. Adding the field to the event data enables you to use the information in visualizations.

```
if [request_http_headers] {
  ruby {
    code => "event.get(['request_http_headers']).collect {|i| event.set(['x_mycompany_employee_num'], i['X-Employee-Num']) if
i.has_key?('X-Employee-Num')}"
  }
}
```

Modifying an existing field

Sometimes you don't want to remove a field entirely from your data, you just want to redact sensitive information such as IDs to prevent them from being exposed in visualizations. You can modify the contents of a field and replace information with a symbols or a message.

Remember: The following fields should not be modified if the data is being written to internal analytics storage: `org_id`, `catalog_id`, `space_id`, `developer_org_id`, `datetime`, and `@timestamp`.

The following example replaces sensitive information in the `X-Employee-Name` and `X-Employee-ID` request headers with the string: `*****sanitized*****`.

```
if [request_http_headers] {
  ruby {
    code => "headers=['X-Employee-Name','X-Employee-ID']; newHeaders = event.get(['request_http_headers']).collect {|i|
headers.each {|header| i[header] = '*****sanitized*****' if i.has_key?(header)}; i}; event.set(['request_http_headers'],
newHeaders)"
  }
}
```

Removing an existing field

Use the mutate `remove_field` operation to delete a field from the Analytics data. To remove multiple fields, delimit the field names with commas.

The following example removes multiple fields (`request_http_headers`, `response_http_headers`, `request_body`, `response_body`, and `query_string`) from the Analytics data:

```
mutate {
  remove_field => ["request_http_headers", "response_http_headers", "request_body", "response_body", "query_string"]
}
```

Filter and output plugin for splunk

When you create a filter for use with splunk, you must also create an output plugin for it.

Filter:

```
if "apicapievent" in [tags] {
  ruby {
    code => "event.set(['@metadata][newevent]', event.to_json)"
  }
}
```

```
}  
}
```

Output plugin:

```
http {  
  url => "http://your-domain/services/collector/event"  
  http_method => "post"  
  codec => "json"  
  content_type => "application/json"  
  id => "offload_http"  
  format => "message"  
  message => '{"event": "%{[@metadata] [newevent]}", "index": "applications-int-ms", "sourcetype": "json:apimlogs"}'  
  headers => ["Authorization", "Splunk XXXX"]  
}
```

Deploying the Analytics subsystem

Define your analytics subsystem, create an ISO file for each analytics node, and deploy the analytics nodes using the Analytics OVA file on a VMware virtual server.

Before you begin

Before you deploy:

- Review the [Deployment requirements on VMware](#).
- Review the [API Connect configuration on VMware](#).
- Review [What's new for v10 installation](#)
- For information on deploying a cluster, see [Configuring API Connect subsystems in a cluster](#)
- If you are upgrading from a previous version, see [Upgrading API Connect](#).

About this task

You must deploy the API Connect OVA template to create each Analytics virtual server that you want in your cloud.

- [Configuring the Analytics subsystem](#)
Specify configuration properties for your Analytics subsystem, and create the ISO files.
- [Deploying the Analytics subsystem OVA file](#)
After you have specified configuration settings and created your ISO files, deploy the subsystem OVA into an appliance VM.
- [Verifying deployment of the Analytics subsystem](#)
Verify that deployment of the Analytics OVA file succeeded.

Configuring the Analytics subsystem

Specify configuration properties for your Analytics subsystem, and create the ISO files.

Before you begin

Review the requirements. See [Deploying the Analytics subsystem](#).

About this task

Use the `apicup` installation utility to specify configuration settings for your Analytics subsystem.

Procedure

1. Ensure that you obtained the distribution file and have a project directory, as described in [First steps for deploying in a VMware environment](#).
2. Change to the project directory.

```
cd myProject
```

3. Create an analytics subsystem.

```
apicup subsys create analyt analytics
```

Where:

- `analyt` is the name of the analytics server that you are creating. You can assign it any name, as long as the identifier consists of lowercase alphanumeric characters or '-', with no spaces, starts with an alphabetic character, and ends with an alphanumeric character.
- `analytics` indicates that you want it to create an Analytics subsystem.

The `apiconnect-up-v10.yml` file that is in that directory is updated to add the analytics-related entries.

Tip: At any time, you can view the current analytics subsystem values in the `apiconnect-up-v10.yml` by running the `apicup get` command:

```
apicup subsys get analyt
```

If you have not updated the value, a default value is listed, if there is one that is available.

Sample output from `apicup subsys`

```
get:
```

Appliance settings

Name	Value	Description
additional-cloud-init-file		(Optional) Path to additional cloud-init yml file
data-device	sdb	VM disk device (usually `sdb` for SCSI or `vdb` for VirtIO)
default-password		(Optional) Console login password for `apicadm` user, password must be pre-hashed
dns-servers	[]	List of DNS servers
extra-values-file		(Optional) Path to additional configuration yml file
k8s-pod-network	172.16.0.0/16	(Optional) CIDR for pods within the appliance
k8s-service-network	172.17.0.0/16	(Optional) CIDR for services within the appliance
public-iface	eth0	Device for API/UI traffic (Eg: eth0)
search-domain	[]	List for DNS search domains
ssh-keyfiles	[]	List of SSH public keys files
traffic-iface	eth0	Device for cluster traffic (Eg: eth0)

Subsystem settings

Name	Value	Description
analytics-backup-auth-pass		(Optional) Server password for analytics backups
analytics-backup-auth-user		(Optional) Server username for analytics backups
analytics-backup-certs		(Optional) Backup certs which are used for TLS communication between APIC and backup server. Currently only supported for s3 backups.
analytics-backup-chunk-size	1GB	analytics-backup-chunk-size
analytics-backup-host		(Optional) FQDN for analytics backups server
analytics-backup-path		(Optional) path for analytics backups server
analytics-backup-schedule	0 0 * * *	(Optional) Cron schedule for analytics backups
analytics-enable-compression	true	(Optional) Determines whether metadata files are stored in compressed format
analytics-enable-server-side-encryption	false	(Optional) Determines whether files are encrypted. When set to true, files are encrypted on the server side using AES256
deployment-profile	n1xc2.m16	Deployment profile (n1xc2.m16/n3xc4.m16) for analytics,
license-use	nonproduction	(n1xc2.m16/n1xc4.m16/n3xc2.m16/n3xc4.m16) for management, (n1xc2.m8/n1xc4.m16/n1xc8.m16/n3xc4.m8/n3xc8.m16) for portal License use (production/nonproduction)

Endpoints

Name	Value	Description
analytics-ingestion		FQDN of Analytics ingestion endpoint

- Specify your `deployment-profile`.

```
apicup subsys set analyt deployment-profile=profile_type
```

Use one of the following values for the `profile_type`, based on the decision you made while [Planning the Analytics deployment profile](#).

For example, to deploy with a three replica profile, run the following command:

```
apicup subsys set analyt deployment-profile=n3xc4.m16
```

Note: The deployment profiles that are shown in the `Description` column of the `apicup get` output are not correct. The available profiles are documented in [Planning your deployment topology and profiles](#).

- Specify the license use that you purchased.

```
apicup subsys set analyt license-use=license_type
```

The `license_type` must be either `production` or `nonproduction`. If not specified, the default value is `nonproduction`.

- Optional: Configure backups of the subsystem. Follow the instructions in [Backing up and restoring the Analytics database on VMware](#). Important: It is highly recommend that you configure backups and also take additional steps to ensure that your configuration and data can be restored in the event of a disaster event. See [Preparing the Analytics subsystem for disaster recovery on VMware](#).

- Optional: Configure your logging.

Logging can be configured at a later time, but you must enable it before installation to capture the log events from the installation.

- Complete the procedure at [Configuring remote logging for a VMware deployment](#).
- Enter the following command to create the log file:

```
apicup subsys set analyt additional-cloud-init-file=config_file.yml
```

- Set your ingestion endpoint.

```
apicup subsys set analyt analytics-ingestion=unique_hostname.domain
```

The ingestion endpoint must be a unique host name that points to the IP address of the OVA (single node deployment), or to the IP of a load balancer configured in front of the OVA nodes. You will use the ingestion endpoint when registering the analytics service in the Cloud Manager UI.

- Set your search domain. Multiple search domains should be separated by commas.

```
apicup subsys set analyt search-domain=your_search_domain
```

Where `your_search_domain` is the domain of your servers, entered in all lowercase. Setting this value ensures that your searches also append these values, which are based on your company's DNS resolution, at the end of the search value. A sample search domain is `mycompany.example.com`.

Ensure that the value for `your_search_domain` is resolved in the system's `/etc/resolv.conf` file to avoid "502" errors when accessing the Cloud Manager web site. For example:

```
# Generated by resolvconf
search your_search_domain ibm.com other.domain.com
```

- Set your domain name servers (DNS).
Supply the IP addresses of the DNS servers for your network. Use a comma to separate multiple server addresses.

```
apicup subsys set analyt dns-servers=ip_address_of_dns_server
```

DNS entries may not be changed on a cluster after the initial installation.

- Set a Public key.

```
apicup subsys set analyt ssh-keyfiles=path_to_public_ssh_keyfile
```

Setting this key enables you to use **ssh** with this key to log in to the virtual machine to check the status of the installation. You will perform this check later in [Verifying deployment of the Analytics subsystem](#).

- You can set the password that you enter to log into your Analytics server for the first time.
 - Review the requirements for creating and using a hashed password. See [Setting and using a hashed default password](#).
 - If you do not have a password hashing utility, install one.

Operating system	Command
Ubuntu, Debian, OSX	If the mkpasswd command utility is not available, download and install it. (You can also use a different password hashing utility.) On OSX, use the command: gem install mkpasswd .
Windows, Red Hat	If necessary, a password hashing utility for the Windows operating system, like OpenSSL

- Create a hashed password

Operating system	Command
Ubuntu, Debian, OSX	mkpasswd --method=sha-512 --rounds=4096 password
Windows, Red Hat	For example, using OpenSSL: openssl passwd -1 password . Note that you might need to add your password hashing utility to your path; for example, on Windows: set PATH=c:\cygwin64\bin;%PATH%

- Set the hashed password for your subsystem:

```
apicup subsys set analyt default-password='hashed_password'
```

Notes:

- The password is hashed. If it is in plain text, you cannot log into the VMWare console.
- Note that the password can only be used to login through the VMWare console. You cannot use it to SSH into the Appliance as an alternative to using the **ssh-keyfiles**.
- On Linux or OSX, use single quotes around *hashed_password*. For Windows, use double quotes.
- If you are using a non-English keyboard, understand the limitations with using the remote VMWare console. See [Requirements for initial deployment on VMWare](#).

- Optional: If the default IP ranges for the API Connect Kubernetes pod and the service networks conflict with IP addresses that must be used by other processes in your deployment, modify the API Connect values.

You can change the IP ranges of the Kubernetes pod and the service networks from the default values of 172.16.0.0/16 and 172.17.0.0/16, respectively. In the case that a /16 subnet overlaps with existing IPs on the network, a Classless Inter-Domain Routing (CIDR) as small as /22 is acceptable. You can modify these ranges during initial installation and configuration only. You cannot modify them once an appliance has been deployed. See [API Connect configuration on VMWare](#).

- Update the IP range for the Kubernetes pod

```
apicup subsys set analyt k8s-pod-network='new_pod_range'
```

Where *new_pod_range* is the new value for the range.

- Update the IP range for Service networks.

```
apicup subsys set analyt k8s-service-network='new_service_range'
```

Where *new_service_range* is the new value for the range.

- Define the hostname for each subsystem node you are deploying:

```
apicup hosts create analyt hostname.domainname hd_password
```

Where the following are true:

- hostname.domainname* is the fully qualified name of the server where you are hosting your Analytics service, including the domain information.
- hd_password* is the password of the Linux Unified Key Setup uses to encrypt the storage for your Analytics service. This password is hashed when it is stored on the server or in the ISO. Note that the password is base64 encoded when stored in **apiconnect-up-v10.yml**.

Repeat this command for each subsystem VM in your deployment. For example, if you are deploying a single node one replica profile then do this once, for a three replica profile you will define three VMs.

Note: Host names and DNS entries may not be changed on a cluster after the initial installation.

- Define the network interface for each subsystem node you are deploying:

```
apicup iface create analyt hostname.domainname physical_network_id host_ip_address/subnet_mask gateway_ip_address
```

Where *physical_network_id* is the network interface ID of your physical server. The value is most often **eth0**. The value can also be **ethx**, where **x** is a number identifier.

The format is similar to this example: **apicup iface create analyt myHostname.domainname eth0 192.0.2.1/255.255.1.1 192.0.2.1**

Repeat this command for each subsystem VM in your deployment. For example, if you are deploying a single node one replica profile then do this once, for a three replica profile you will define three VMs.

16. Optional: Use **apicup** to view the configured hosts:

```
apicup hosts list analyt
testsrv0233.subnet1.example.com
  Device IP/Mask Gateway
  eth0 1.2.152.233/255.255.254.0 1.2.152.1
```

17. Optional: Enable JWT security instead of mTLS for communication from management and gateway to your analytics subsystem. JWT security provides application layer security and can be used instead of mTLS when there are load-balancers located between subsystems that require TLS termination. For more information about JWT security, see [Enable JWT instead of mTLS](#). To enable JWT and disable mTLS, first identify the JWKS URL from the management subsystem:

```
apicup subsys get <management subsystem>
...
jwks-url https://appliance1.apic.acme.com/api/cloud/oauth2/certs JWKS URL for Portal and Analytics subsystems to
validate JWT -- this is unsettable and is generated based on the platform-api endpoint
...
```

Disable mTLS and enable JWT by setting the jwks-url with **apicup**:

```
apicup subsys set analyt mtls-validate-client=false
apicup subsys set analyt jwks-url=https://appliance1.apic.acme.com/api/cloud/oauth2/certs
```

JWT for gateway to analytics communication requires an additional step during gateway registration. Enable the Use JWT switch when you register the gateway in the Cloud Manager UI.

Note: Do not disable mTLS without enabling JWT.

18. Verify that the configuration settings are valid.

```
apicup subsys get analyt --validate
```

The output lists each setting and adds a check mark after the value once the value is validated. If the setting lacks a check mark and indicates an invalid value, reconfigure the setting. See the following sample output.

```
apicup subsys get analyt --validate
Appliance settings
=====
Name Value
----
additional-cloud-init-file ✓
data-device sdb ✓
default-password $6$rounds=4096$iMCJ9cFhFJ8X$pbmAl9ClWzcYzH
ZFoQ6n7OnYcf/owQZiIcPAtWazs/FUn/uE8uLD.9jwHE0AX4upFSqx/jf0ZmDbHPZ9bU1CY1 ✓
dns-servers [1.2.3.1] ✓
extra-values-file ✓
k8s-pod-network 172.16.0.0/16 ✓
k8s-service-network 172.17.0.0/16 ✓
public-iface eth0 ✓
search-domain [subnet1.example.com] ✓
ssh-keyfiles [/home/vsphere/.ssh/id_rsa.pub] ✓
traffic-iface eth0 ✓

Subsystem settings
=====
Name Value
----
analytics-backup-auth-pass ✓
analytics-backup-auth-user ✓
analytics-backup-certs ✓
analytics-backup-chunk-size 1GB ✓
analytics-backup-host ✓
analytics-backup-path ✓
analytics-backup-schedule 0 0 * * * ✓
analytics-enable-compression true ✓
analytics-enable-server-side-encryption false ✓
deployment-profile n1xc2.m16 ✓
license-use production ✓

Endpoints
=====
Name Value
----
analytics-ingestion a7s-in.testsrv0233.subnet1.example.com ✓
```

19. If you configured additional deployment options in an extra-values file, run the following command to make the file available during installation:

```
apicup subsys set analyt extra-values-file path/analytics-extra-values.yaml
```

where **analytics-extra-values.yaml** is the name of the file containing your deployment settings.

20. Create your ISO files:

```
apicup subsys install analyt --out analytplan-out
```

The **--out** parameter and value are required. In this example, the ISO files are created in the myProject/analytplan-out directory.

There will be one ISO file for each VM host you defined in step [14](#).

If the system cannot find the path to your software that creates ISO files, create a path setting to that software by running a command similar to the following command:

Operating system	Command
Ubuntu, Debian, OSX	<code>export PATH=\$PATH:/Users/your_path/</code>
Windows, Red Hat	<code>set PATH="c:\Program Files (x86)\cdrtools";%PATH%</code>

21. Continue with [Deploying the Analytics subsystem OVA file](#).

Deploying the Analytics subsystem OVA file

After you have specified configuration settings and created your ISO files, deploy the subsystem OVA into an appliance VM.

About this task

Repeat this procedure for each analytics VM you are deploying. Use the ISO files created in [Configuring the Analytics subsystem](#).

Procedure

1. Log into the VMWare vSphere Web Client.
2. Using the VSphere Navigator, navigate to the directory where you are deploying the OVA file.
3. Right-click the directory and select Deploy OVF Template.
4. Complete the Deploy OVF Template wizard.
 - a. Select the Analytics subsystem .OVA file template by navigating to the location where you downloaded the file in [First steps for deploying in a VMware environment](#).
 - b. Enter a name and location for your file.
 - c. Select a resource for your template.
 - d. Review the details for your template.
 - e. Select the size of your configuration.
 - f. Select the storage settings.
Note: Ensure you specify a storage size large enough for your estimated analytics data usage: [Estimating internal storage space](#)
 - g. Select the networks.
 - h. Customize the Template, if necessary.
 - i. Review the details to ensure that they are correct.
 - j. Select Finish to deploy the virtual machine.

Note: Do not change the OVA hardware version, even if the VMware UI shows a Compatibility range that includes other versions. See [Requirements for initial deployment on VMware](#).

The template creation appears in your Recent Tasks list.
5. Select the Storage tab in the Navigator.
6. Navigate to your datastore.
7. Upload your ISO file.
 - a. Select the Navigate to the datastore file browser icon in the icon menu.
 - b. Select the Upload a file to the Datastore icon in the icon menu.
 - c. Navigate to the ISO file that you created in your project.
It is the myProject/analytplan-out
 - d. Upload the ISO file to the datastore.
8. Leave the datastore by selecting the VMs and Templates icon in the Navigator.
9. Locate and select your virtual machine.
10. Select the Configure tab in the main window.
11. Select Edit....
 - a. On the Virtual Hardware tab, select CD/DVD Drive 1.
 - b. For the Client Device, select Datastore ISO File.
 - c. Find and select your datastore in the Datastores category.
 - d. Find and select your ISO file in the Contents category.
 - e. Select OK to commit your selection and exit the Select File window.
 - f. Ensure that the Connect At Power On check box is selected.
Tip:
 - Expand the CD/DVD drive 1 entry to view the details and the complete Connect At Power On label.
 - Note that VMware related issues with ISO mounting at boot may occur if Connect At Power On
 - g. Select OK to commit your selection and close the window.
12. Start the virtual machine by selecting the play button on the icon bar.
The installation might take several minutes to complete, depending on the availability of the system and the download speed.
13. Next, verify the deployment. Continue with [Verifying deployment of the Analytics subsystem](#).

Verifying deployment of the Analytics subsystem

Verify that deployment of the Analytics OVA file succeeded.

Before you begin

You must have completed [Deploying the Analytics subsystem OVA file](#).

About this task

After you configure an ISO and deploy it with the Analytics subsystem OVA, as described in [Deploying the Analytics subsystem OVA file](#), verify that the subsystem installed correctly and is functional.

Procedure

1. Log in to the virtual machine by using an SSH tool to check the status of the installation:
 - a. Enter the following command to connect to *analyt* using SSH:

```
ssh ip_address -l apicadm
```

You are logging in with the default ID of *apicadm*, which is the API Connect ID that has administrator privileges.

- b. Select Yes to continue connecting.
Your host names are automatically added to your list of hosts.
- c. Run the **apic status** command to verify that the installation completed and the system is running correctly.
The command output for a correctly running Analytics system is similar to the following:

```
#sudo apic status
INFO[0000] Log level: info

Cluster members:
- apimdev0239.subnet1.example.com (9.20.152.239)
  Version: 4.4.1
  Type: BOOTSTRAP_MASTER
  Install stage: DONE
  Upgrade stage: UPGRADE_DONE
  Subsystem type: analytics
  Subsystem version: 10.0.5.1
  Subsystem detail:
  Reboot Required: false
  Containerd status:
    Systemd unit: running
  Kubernetes status:
    Systemd unit: running
    Kubelet version: apimdev0239 (5.4.0-122-generic) [Kubelet v1.23.5, Proxy v1.23.5]
  Etcd status: pod etcd-apimdev0239 in namespace kube-system has status Running
  Addons: calico, cert-manager-v1, dns, kube-proxy, local-volume-provisioner, metrics-server,
  Disk Space Available: 226G/245G | Used: 4%
  Inodes Available: 16M/16M | Used: 1%
  Containers: 21
    - Running: 21
    - Exited: 7
    - Unknown: 0
  Images: 32
Etcd cluster state:
- etcd member name: apimdev0239.subnet1.example.com, member id: 15210952383217851221, cluster id:
8720375562272123817, leader id: 15210952383217851221, revision: 2538453, version: 3.5.4
```

Pods Summary:

NODE	NAMESPACE	NAME	READY	STATUS
apimdev0239	certmanager	cert-manager-84f999d4c7-jz8p6	1/1	Running
apimdev0239	certmanager	cert-manager-cainjector-5dc89f744d-7mbms	1/1	Running
apimdev0239	certmanager	cert-manager-webhook-7497758675-vn4hp	1/1	Running
apimdev0239	default	cpd-analytics-director-7d584d8ffd-qts9c	1/1	Running
apimdev0239	default	cpd-analytics-ingestion-0	1/1	Running
apimdev0239	default	cpd-analytics-mtls-gw-f647687d6-9rbpd	1/1	Running
apimdev0239	default	cpd-analytics-osinit-md7cn	0/1	Succeeded
apimdev0239	default	cpd-analytics-storage-0	1/1	Running
apimdev0239	default	ibm-apiconnect-8cd7cb6bd-vjwv2	1/1	Running
apimdev0239	kube-system	calico-kube-controllers-789ff9469b-xrttr	1/1	Running
apimdev0239	kube-system	calico-node-8tfrx	1/1	Running
apimdev0239	kube-system	coredns-79f7dbf799-7dmn6	1/1	Running
apimdev0239	kube-system	coredns-79f7dbf799-9qgnk	1/1	Running
apimdev0239	kube-system	etcd-apimdev0239	1/1	Running
apimdev0239	kube-system	ingress-nginx-ingress-controller-96k8k	1/1	Running
apimdev0239	kube-system	kube-apiserver-apimdev0239	1/1	Running
apimdev0239	kube-system	kube-apiserver-proxy-apimdev0239	1/1	Running
apimdev0239	kube-system	kube-controller-manager-apimdev0239	1/1	Running
apimdev0239	kube-system	kube-proxy-j41bz	1/1	Running
apimdev0239	kube-system	kube-scheduler-apimdev0239	1/1	Running
apimdev0239	kube-system	local-volume-provisioner-rlfjs	1/1	Running
apimdev0239	kube-system	metrics-server-d9fffc9-nmcgr	1/1	Running

2. If you have now installed all subsystems, continue to [Post-deployment steps](#).

Deploying the Developer Portal virtual appliance

You create a Developer Portal node by deploying the Developer Portal OVA template. After you deploy the Developer Portal OVA template, you can install the Developer Portal.

Before you begin

Before you deploy:

- Review the [Deployment requirements on VMware](#).
- Review the [API Connect configuration on VMware](#).
- Review [What's new for v10 installation](#)
- For information on deploying a cluster, see [Configuring API Connect subsystems in a cluster](#)
- If you are upgrading from a previous version, see [Upgrading API Connect](#).

Note:

Ensure that your kernel or Kubernetes node has the value of its `inotify` watches set high enough so that the Developer Portal can monitor and maintain the files for each Developer Portal site. If set too low, the Developer Portal containers might fail to start or go into a `non-ready` state when this limit is reached. If you have many Developer Portal sites, or if your sites contain a lot of content, for example, many custom modules and themes, then a larger number of `inotify` watches are required. You can start with a value of 65,000, but for large deployments, this value might need to go up as high as 1,000,000. The Developer Portal containers take `inotify` watches only when they need them. The full number is not reserved or held, so it is acceptable to set this value high.

About this task

You must deploy the Developer Portal OVA template to create each Developer Portal node that you want in your cloud. Each node has a separate CLI password account that is required to log in through a Secure Shell (SSH) to complete specific administrative actions for only that node.

Important deployment information for Developer Portal:

- You must deploy the Developer Portal OVA template by using a version of the VMware vSphere Client that supports the SHA-512 Cryptographic Hash Algorithm.
- The Developer Portal node is initially configured with a default password of `7iron-hide`, with the user name of `admin`. For security reasons, change the default password by completing one of the following actions:
 - During deployment, if the feature is available in your VMware instance, enter a new password. If you specify a password during deployment, the password for the Developer Portal command line interface (CLI) is modified. Use the new password when you log into the CLI. You cannot modify the admin user name for the CLI.
 - After deployment, log in to the CLI for each virtual appliance and run the command to change the default password for that specific node. The CLI command is `passwd`.

Note that this console uses a US keyboard configuration, in which the `@` symbol can be in a different place to other keyboard configurations. If you are not using a US keyboard, ensure that you typed the password correctly.

- Only static IP addresses that are specified during the `apicup` project configuration before the installation of the OVAs are supported.
- To enable effective high availability for your Portal service, you need a latency that is less than 50ms between all OVAs to avoid the risk of performance degradation. Servers with uniform specifications are required, as any write actions occur at the speed of the slowest OVA, as the write actions are synchronous across the cluster of OVAs. It is recommended that there are three servers in each cluster of OVAs for the high availability configuration. The three servers can be situated in the same data center, or across three data centers to ensure the best availability. However, you can configure high availability with two data centers.

Procedure

Complete the following steps in order:

Note: The following steps apply to VMware only. Depending on the VMware version that you are using, some of the steps might vary. For example, you might not be able to change the user name and password during deployment.

1. Specify configuration settings for the Developer Portal subsystem and create an ISO. See [Configuring the Developer Portal subsystem](#)
2. Deploy the subsystem OVA into an appliance VM. See [Deploying the Developer Portal subsystem OVA file](#).
3. Verify the deployment. See [Verifying deployment of the Developer Portal subsystem](#)
4. Optional: Define multiple Developer Portal endpoints. See [Defining multiple portal endpoints for a VMware environment](#)

What to do next

If you want to deploy an API Connect Analytics OVA file, continue with [Deploying the Analytics subsystem](#).

If you did not specify a new password during deployment in VMware, then after deployment, log in to the command-line interface (CLI) for each appliance and run the command `passwd` to change the password.

- [Configuring the Developer Portal subsystem](#)
Specify configuration properties for your Developer Portal subsystem, and create an ISO.
- [Deploying the Developer Portal subsystem OVA file](#)
After you have specified configuration settings and created an ISO, deploy the subsystem OVA into an appliance VM.
- [Verifying deployment of the Developer Portal subsystem](#)
Verify that deployment of the Developer Portal OVA file succeeded.
- [Configuring two NICs on the Developer Portal](#)
Configure the Developer Portal to use two network interface cards.
- [Specifying a range of allowable client IP addresses for Developer Portal](#)
Specify the range of client IP addresses to be allowed in a deployment of the Developer Portal
- [Defining multiple portal endpoints for a VMware environment](#)
Multiple public facing endpoints (`portal-xxx`) can be defined for the Developer Portal.

Configuring the Developer Portal subsystem

Specify configuration properties for your Developer Portal subsystem, and create an ISO.

Before you begin

Review the requirements and important deployment information for Developer Portal. See [Deploying the Developer Portal virtual appliance](#).

About this task

Use the `apicup` installation utility to specify configuration settings for your Developer Portal subsystem.

Procedure

1. Ensure that you obtained the distribution file and have a project directory, as described in [First steps for deploying in a VMware environment](#).
2. Change to the project directory.

```
cd myProject
```

3. Create a portal subsystem.

```
apicup subsys create port portal
```

Where:

- `port` is the name of the Developer Portal server that you are creating. You can assign it any name, as long as the identifier consists of lowercase alphanumeric characters or '-', with no spaces, starts with an alphabetic character, and ends with an alphanumeric character.
- `portal` indicates that you want it to create a Developer Portal microservice.

Note: Unless you are installing a two data center deployment, leave `multi-site-ha-enabled` as false, and ignore `multi-site-ha-mode`, `replication-peer-fqdn`, `site-name`, and the `portal-replication` endpoint, as these options are all related to that feature. For more information, see [Installing a two data center deployment on VMware](#).

The `apiconnect-up-v10.yml` file that is in that directory is updated to add the portal-related entries.

Tip: At any time, you can view the current developer portal subsystem values in the `apiconnect-up-v10.yml` by running the `apicup get` command:

```
apicup subsys get port
```

If you have not yet configured the subsystem, the command might return errors. Also, if you have not updated the value, a default value is listed, if there is one that is available.

After configuration is complete, you can view output similar to the following sample:

```
Appliance settings
=====
Name                               Value
Description                         -----
-----
additional-cloud-init-file
(Optional) Path to additional cloud-init yml file
data-device                          sdb
VM disk device (usually `sdb` for SCSI or `vdb` for VirtIO)
default-password
$6$rounds=4096$vtcqpAVK$dzqrOeYP33WTvTug38Q4Rld518TmdQgezzTnkX/PFwkzTiZ2S0CqNRr1S4b08tOc4p.OEg4BtzBe/r8RAk.gw/
(Optional) Console login password for `apicadm` user, password must be pre-hashed
dns-servers                          [192.168.1.201]
List of DNS servers
extra-values-file
(Optional) Path to additional configuration yml file
k8s-pod-network                      172.16.0.0/16
(Optional) CIDR for pods within the appliance
k8s-service-network                 172.17.0.0/16
(Optional) CIDR for services within the appliance
public-iface                         eth0
Device for API/UI traffic (Eg: eth0)
search-domain                       [subnet1.example.com]
List for DNS search domains
ssh-keyfiles                         [/home/vsphere/.ssh/id_rsa.pub]
List of SSH public keys files
traffic-iface                        eth0
Device for cluster traffic (Eg: eth0)

Subsystem settings
=====
Name                               Value
Description                         -----
-----
deployment-profile                  n1xc2.m8
Deployment profile (n1xc2.m16/n3xc4.m16) for analytics, (n1xc4.m16/n3xc4.m16) for management, (n1xc2.m8/n3xc4.m8) for
portal
license-use                         nonproduction
License use (production/nonproduction)
multi-site-ha-enabled                false
Multi site HA enabled
multi-site-ha-mode                   active
Multi site HA mode (active/passive)
replication-peer-fqdn
Replication peer fully qualified name (replication endpoint of active mode site)
site-name
Site name, used in k8s resource names

Endpoints
=====
Name                               Value
Description                         -----
```

```

-----
portal-admin                portal-api.example.com
FQDN of Portal admin endpoint
portal-replication
FQDN of Portal replication endpoint, only required if HA is enabled
portal-www                  portal-www.example.com
FQDN of Portal web endpoint

```

4. Specify **deployment-profile**.

```
apicup subsys set port deployment-profile=n3xc4.m8
```

The **deployment-profile** parameter indicates that you are deploying a three replica or one replica environment. For more information, see [Planning your deployment topology and profiles](#).

Note: The deployment profiles that are shown in the **Description** column of the **apicup get** output are not correct. The available profiles are documented in [Planning your deployment topology and profiles](#).

5. Specify the license version you purchased.

```
apicup subsys set port
license-use=<license_type>
```

The *license_type* must be either **production** or **nonproduction**. If not specified, the default value is **nonproduction**.

6. Optional: Configure scheduled backups of the subsystem. This step is optional but is recommended.

Refer to the instructions in [Backing up and restoring the Developer Portal](#).

7. Optional: Configure your logging.

Logging can be configured at a later time, but you must enable it before installation to capture the log events from the installation.

- Complete the procedure at [Configuring remote logging for a VMware deployment](#).
- Enter the following command to create the log file:

```
apicup subsys set port additional-cloud-init-file=config_file.yml
```

8. Enter the following commands to update the apiconnect-up-v10.yml with the information for your environment:

- Use **apicup** to set your endpoints.

You can use wildcard aliases or host aliases with your endpoints.

Optionally, you can specify all endpoints with one **apicup** command.

Note: You cannot specify the underscore character "_" in domain names that are used in endpoints. See [API Connect configuration on VMware](#).

The endpoints must be unique hostnames which both point to the IP address of the OVA (single node deployment), or to the IP of a load balancer configured in front of the OVA nodes. See examples in the sample output in step 3.

Setting	Endpoint host description
portal-admin	This is the <i>unique_hostname</i> for communication between your Cloud Manager and API Manager, and your Developer Portal. The values for the portal-admin and portal-www must be different. <pre>apicup subsys set port portal-admin=unique_hostname.domain</pre>
portal-www	This is the <i>unique_hostname</i> for the Developer Portal Internet site that is created for the Developer Portal. Multiple portal-www endpoints may be configured, as described here: Defining multiple portal endpoints for a VMware environment . <pre>apicup subsys set port portal-www=unique_hostname.domain</pre>

- Set your search domain. Multiple search domains should be separated by commas.

```
apicup subsys set port search-domain=your_search_domain
```

Where *your_search_domain* is the domain of your servers, entered in all lowercase. Setting this value ensures that your searches also append these values, which are based on your company's DNS resolution, at the end of the search value. A sample search domain is *mycompany.example.com*.

Ensure that the value for *your_search_domain* is resolved in the system's `/etc/resolv.conf` file to avoid "502" errors when accessing the Cloud Manager web site. For example:

```
# Generated by resolvconf
search your_search_domain ibm.com other.domain.com
```

- Set your domain name servers (DNS).

Supply the IP addresses of the DNS servers for your network. Use a comma to separate multiple server addresses.

```
apicup subsys set port dns-servers=ip_address_of_dns_server[,ip_address_of_another_dns_server_if_necessary]
```

DNS entries may not be changed on a cluster after the initial installation.

9. Set a Public key.

```
apicup subsys set port ssh-keyfiles=path_to_public_ssh_keyfile
```

Setting this key enables you to use **ssh** with this key to log in to the virtual machine to check the status of the installation. You will perform this check in [Verifying deployment of the Developer Portal subsystem](#).

10. If there are multiple Management subsystems in the same project, set the Portal subsystem's **platform-api** and **consumer-api** certificates to match those used by the appropriate Management subsystem to ensure that the Portal subsystem is correctly associated with that Management subsystem.

This step only applies if you installed more than one Management subsystem into a single project.

A Portal subsystem can be associated with only one Management subsystem. To associate the new Portal subsystem with the appropriate Management subsystem, manually set the **mgmt-platform-api** and **mgmt-consumer-api** certificates to match the ones used by the Management subsystem.

- Run the following commands to get the certificates from the Management subsystem:

```
apicup certs get mgmt-subsystem-name platform-api -t cert > platform-api.crt
apicup certs get mgmt-subsystem-name platform-api -t key > platform-api.key
apicup certs get mgmt-subsystem-name platform-api -t ca > platform-api-ca.crt
```

```

apicup certs get mgmt-subsystem-name consumer-api -t cert > consumer-api.crt
apicup certs get mgmt-subsystem-name consumer-api -t key > consumer-api.key
apicup certs get mgmt-subsystem-name consumer-api -t ca > consumer-api-ca.crt

```

where `mgmt-subsystem-name` is the name of the specific Management subsystem that you want to associate the new Portal subsystem with.

- b. Run the following commands to set the Portal's certificates to match those used by the Management subsystem:

```

apicup certs set ptl-subsystem-name mgmt-platform-api Cert_file_path Key_file_path CA_file_path
apicup certs set ptl-subsystem-name mgmt-consumer-api Cert_file_path Key_file_path CA_file_path

```

For more information on `apicup` certificate commands, see [Command reference](#).

11. You can set a hashed password that you enter to log in to your Developer Portal server for the first time.

- a. **Important:** Review the requirements for creating and using a hashed password. See [Setting and using a hashed default password](#).
b. If you do not have a password hashing utility, install one.

Operating system	Command
Ubuntu, Debian, OSX	If the <code>mkpasswd</code> command utility is not available, download and install it. (You can also use a different password hashing utility.) On OSX, use the command: <code>gem install mkpasswd</code> .
Windows, Red Hat	If necessary, a password hashing utility for the Windows operating system, like OpenSSL

- c. Create a hashed password

Operating system	Command
Ubuntu, Debian, OSX	<code>mkpasswd --method=sha-512 --rounds=4096 password</code>
Windows, Red Hat	For example, using OpenSSL: <code>openssl passwd -1 password</code> . Note that you might need to add your password hashing utility to your path; for example, on Windows: <code>set PATH=c:\cygwin64\bin;%PATH%</code>

- d. Set the hashed password for your subsystem:

```

apicup subsys set port default-password="hashed_password"

```

Notes:

- The password is hashed. If it is in plain text, you cannot log into the VMware console.
- Note that the password can only be used to login through the VMware console. You cannot use it to SSH into the Appliance as an alternative to using the `ssh-keyfiles`.
- On Linux or OSX, use single quotes around `hashed_password`. For Windows, use double quotes.
- If you are using a non-English keyboard, understand the limitations with using the remote VMware console. See [Requirements for initial deployment on VMware](#).

12. Optional: If the default IP ranges for the API Connect Kubernetes pod and the service networks conflict with IP addresses that must be used by other processes in your deployment, modify the API Connect values.

You can change the IP ranges of the Kubernetes pod and the service networks from the default values of 172.16.0.0/16 and 172.17.0.0/16, respectively. In the case that a /16 subnet overlaps with existing IPs on the network, a Classless Inter-Domain Routing (CIDR) as small as /22 is acceptable. You can modify these ranges during initial installation and configuration only. You cannot modify them once an appliance has been deployed. See [API Connect configuration on VMware](#).

- a. Update the IP range for the Kubernetes pod

```

apicup subsys set port k8s-pod-network='new_pod_range'

```

Where `new_pod_range` is the new value for the range.

- b. Update the IP range for Service networks.

```

apicup subsys set port k8s-service-network='new_service_range'

```

Where `new_service_range` is the new value for the range.

13. Add your hosts.

```

apicup hosts create port hostname.domainname hd_password

```

Where the following are true:

- `hostname.domainname` is the fully qualified name of the server where you are hosting your Developer Portal, including the domain information.
- `hd_password` is the password of the Linux Unified Key Setup uses to encrypt the storage for your Developer Portal. This password is hashed when it is stored on the server or in the ISO. Note that the password is base64 encoded when stored in `apiconnect-up-v10.yml`. Repeat this command for each host that you want to add.

Note: Host names and DNS entries may not be changed on a cluster after the initial installation.

14. Create your interfaces.

```

apicup iface create port hostname.domainname physical_network_id host_ip_address/subnet_mask gateway_ip_address

```

Where `physical_network_id` is the network interface ID of your physical server. The value is most often `eth0`. The value can also be `ethx`, where `x` is a number identifier.

The format is similar to this example: `apicup iface create port myHostname.domain eth0 192.0.2.1/255.255.1.1 192.0.2.1`

Note: You can optionally configure a second network interface (NIC) card for use with Developer Portal. In this scenario, one NIC is used for internal traffic between the Portal and APIM, and the second is used as a public interface. If you are creating multiple network interfaces, each one must be on a different subnet with a different gateway. For information on configuring two NICs for Developer Portal, see [Configuring two NICs on the Developer Portal](#).

15. Optional: You can specify a range of allowable client IP addresses.

You can optionally restrict which IP addresses can access any of the ingresses by creating an allowlist annotation, which then will only allow the specified IP addresses to access the ingress and will deny all other source IP addresses. See [Specifying a range of allowable client IP addresses for Developer Portal](#).

Note: One usage of this restriction is the scenario where you split traffic between your private and public networks in 2 NIC configuration.

16. Optional: Use **apicup** to view the configured hosts:

```
apicup hosts list port
apimdev0232.hursley.ibm.com
  Device IP/Mask Gateway
  eth0 1.2.152.232/255.255.254.0 1.2.152.1
```

Note: If you are configuring a two data center deployment, continue with the installation instructions in [Installing a two data center deployment on VMware](#).

17. Optional: Enable JWT security instead of mTLS for communication from management to your portal subsystem.

JWT security provides application layer security and can be used instead of mTLS when there are load-balancers located between subsystems that require TLS termination. For more information about JWT security, see [Enable JWT instead of mTLS](#). To enable JWT and disable mTLS, first identify the JWKS URL from the management subsystem:

```
apicup subsys get <management subsystem>
...
jwks-url https://appliance1.apic.acme.com/api/cloud/oauth2/certs JWKS URL for Portal and Analytics subsystems to
validate JWT -- this is unsettable and is generated based on the platform-api endpoint
...
```

Disable mTLS and enable JWT by setting the jwks-url with **apicup**:

```
apicup subsys set port mTLS-validate-client=false
apicup subsys set port jwks-url=https://appliance1.apic.acme.com/api/cloud/oauth2/certs
```

Note: Do not disable mTLS without enabling JWT.

18. Leave **multi-site-ha-enabled** set to **false**.

19. Optional: Verify that the configuration settings are valid.

```
apicup subsys get port --validate
```

The output lists each setting and adds a check mark after the value once the value is validated. If the setting lacks a check mark and indicates an invalid value, reconfigure the setting. See the following sample output.

```
Appliance settings
=====
Name Value
----
additional-cloud-init-file
✓
data-device sdb
✓
default-password
$6$rounds=4096$vtcqpAVK$dzqrOeYP33WTVtug38Q4Rld5l8TmdQgezzTnkX/PFwkzTiZ2S0CqNRr1S4b08tOc4p.OEg4BtzBe/r8RAk.gW/ ✓
dns-servers [192.168.1.201]
✓
extra-values-file
✓
k8s-pod-network 172.16.0.0/16
✓
k8s-service-network 172.17.0.0/16
✓
public-iface eth0
✓
search-domain [subnet1.example.com]
✓
ssh-keyfiles [/home/vsphere/.ssh/id_rsa.pub]
✓
traffic-iface eth0
✓

Subsystem settings
=====
Name Value
----
deployment-profile nlxc2.m8
✓
license-use nonproduction
✓
multi-site-ha-enabled false
✓
multi-site-ha-mode active
✓
replication-peer-fqdn
✓
site-name
✓

Endpoints
=====
Name Value
```

```

----
portal-admin          portal-api.example.com
✓
portal-replication
✓
portal-www           portal-www.example.com
✓

```

20. Create your ISO file.

```
apicup subsys install port --out portplan-out
```

The `--out` parameter and value are required. In this example, the ISO file is created in the `myProject/portplan-out` directory.

If the system cannot find the path to your software that creates ISO files, create a path setting to that software by running a command similar to the following command:

Operating system	Command
Ubuntu, Debian, OSX	<code>export PATH=\$PATH:/Users/your_path/</code>
Windows, Red Hat	<code>set PATH="c:\Program Files (x86)\cdrtools";%PATH%</code>

21. Next, use the ISO to deploy the subsystem OVA. Continue with [Deploying the Developer Portal subsystem OVA file](#).

Deploying the Developer Portal subsystem OVA file

After you have specified configuration settings and created an ISO, deploy the subsystem OVA into an appliance VM.

Before you begin

You must have completed the configuration of the project file entries for the Developer Portal subsystem, in [Configuring the Developer Portal subsystem](#).

Procedure

1. Log into the VMWare vSphere Web Client.
2. Using the VSphere Navigator, navigate to the directory where you are deploying the OVA file.
3. Right-click the directory and select Deploy OVF Template.
4. Complete the Deploy OVF Template wizard.
 - a. Select the Portal subsystem .OVA file template by navigating to the location where you downloaded the file in [First steps for deploying in a VMware environment](#).
 - b. Enter a name and location for your file.
 - c. Select a resource for your template.
 - d. Review the details for your template.
 - e. Select the size of your configuration.
 - f. Select the storage settings.

Note that the number of Central Processing Units, RAM, and size of disk that you need for the Developer Portal varies depending on the number of sites that are hosted and the number of concurrent users you expect your site to have:

Table 1. Developer Portal hardware requirements

Number of sites	Number of concurrent users	Number of CPUs	Amount of RAM (GB)	Data Disk Size (GB)++
1	1	2**	4	50
20	5	4	16	70
100	20	8	32	100
100	100	16	64	500

Important:

- ++The data disk size is extra to the main disk of the OVA. The main disk of the OVA is sized at 100GB, and should not be changed. Therefore, the total disk size of the OVA is 100GB plus the data disk size. The default data disk size is 200GB. If you want to select a different value, then you need to do this by using the OVF Tool, or the VMware GUI, before you power on the VM. Note that certain versions of the VMware GUI may not allow you to resize the data disk, and in this case you should use the OVF Tool.
- **The requirement of 2 CPUs is suitable only for proof-of-concept work, and non-high availability deployments. For example, this configuration is suitable for the demo mode of the apicup installation, which is set by using the command `apicup subsys set port deployment-profile=n1xc2.m8`.
- It's not recommended to have more than 100 sites per Developer Portal service. Note that it's not necessary to have a Portal site for every Catalog, for example Catalogs that are only for API Developers don't need a Portal site, as the APIs can be tested by using credentials from the API Manager. If more than 100 sites are required, you should configure additional Developer Portal services; see [Registering a Portal service](#).

- g. Select the networks.
- h. Customize the Template, if necessary.
- i. Review the details to ensure that they are correct.
- j. Select Finish to deploy the virtual machine.

Note: Do not change the OVA hardware version, even if the VMware UI shows a Compatibility range that includes other versions. See [Requirements for initial deployment on VMware](#).

The template creation appears in your Recent Tasks list.

5. Select the Storage tab in the Navigator.
6. Navigate to your datastore.
7. Upload your ISO file.
 - a. Select the Navigate to the datastore file browser icon in the icon menu.
 - b. Select the Upload a file to the Datastore icon in the icon menu.
 - c. Navigate to the ISO file that you created in your project.

- It is the myProject/portplan-out
- d. Upload the ISO file to the datastore.
 8. Leave the datastore by selecting the VMs and Templates icon in the Navigator.
 9. Locate and select your virtual machine.
 10. Select the Configure tab in the main window.
 11. Select Edit....
 - a. On the Virtual Hardware tab, select CD/DVD Drive 1.
 - b. For the Client Device, select Datastore ISO File.
 - c. Find and select your datastore in the Datastores category.
 - d. Find and select your ISO file in the Contents category.
 - e. Select OK to commit your selection and exit the Select File window.
 - f. Ensure that the Connect At Power On check box is selected.

Tip:

 - Expand the CD/DVD drive 1 entry to view the details and the complete Connect At Power On label.
 - Note that VMware related issues with ISO mounting at boot may occur if Connect At Power On
 - g. Select OK to commit your selection and close the window.
 12. Start the virtual machine by selecting the play button on the icon bar.
The installation might take several minutes to complete, depending on the availability of the system and the download speed.
 13. Continue with [Verifying deployment of the Developer Portal subsystem](#).

Verifying deployment of the Developer Portal subsystem

Verify that deployment of the Developer Portal OVA file succeeded.

Before you begin

You must have completed [Deploying the Developer Portal subsystem OVA file](#).

About this task

After you configure an ISO and deploy it with the Developer Portal subsystem OVA, as described in [Deploying the Developer Portal subsystem OVA file](#), verify that the subsystem installed correctly and is functional.

Procedure

1. Log in to the virtual machine by using an SSH tool to check the status of the installation:
 - a. Enter the following command to connect to *port* using SSH:

```
ssh ip_address -l apicadm
```

You are logging in with the default ID of *apicadm*, which is the API Connect ID that has administrator privileges.

- b. Select Yes to continue connecting.
Your host names are automatically added to your list of hosts.
- c. Run the **apic status** command to verify that the installation completed and the system is running correctly.
The command output for a correctly running Developer Portal system is similar to the following lines:

```
$ sudo apic status
INFO[0000] Log level: info

Cluster members:
- testsrv1251.subnet1.example.com (1.2.3.4)
  Type: BOOTSTRAP_MASTER
  Install stage: DONE
  Upgrade stage: NONE
  Docker status:
  Systemd unit: running
  Kubernetes status:
  Systemd unit: running
  Kubelet version: testsrv1251 (4.4.0-137-generic) [Kubelet v1.10.6, Proxy v1.10.6]
  Etd status: pod etcd-testsrv1251 in namespace kube-system has status Running
  Addons: calico, dns, helm, kube-proxy, metrics-server, nginx-ingress,
  Etcd cluster state:
- etcd member name: testsrv1251.subnet1.example.com, member id: 10293853252850049269, cluster id:
17044377177359475136, leader id: 10293853252850049269, revision: 1485879, version: 3.1.17
```

Pods Summary:

NODE	NAMESPACE	NAME	READY
testsrv1251	default	re702738954-apic-portal-db-f89vn	2/2
Running	default	re702738954-apic-portal-nginx-6ffb8676d9-gtfc6	0/0
Pending	default	re702738954-apic-portal-nginx-6ffb8676d9-nqdzf	0/0
Pending	default	re702738954-apic-portal-nginx-6ffb8676d9-q85mt	1/1
testsrv1251	default	re702738954-apic-portal-www-p9bvxx	2/2
Running	default	re702738954-apic-portal-www-p9bvxx	2/2
testsrv1251	kube-system	calico-node-xkpbk	2/2
Running	kube-system	calico-node-xkpbk	2/2
testsrv1251	kube-system	coredns-87cb95869-p4qhf	1/1
Running	kube-system	coredns-87cb95869-p4qhf	1/1

testsrv1251	kube-system	coredns-87cb95869-z2n5z	1/1
Running			
testsrv1251	kube-system	etcd-testsrv1251	1/1
Running			
testsrv1251	kube-system	ingress-nginx-ingress-controller-dsnxw	1/1
Running			
testsrv1251	kube-system	ingress-nginx-ingress-default-backend-6f58fb5f56-ldx7t	1/1
Running			
testsrv1251	kube-system	kube-apiserver-testsrv1251	1/1
Running			
testsrv1251	kube-system	kube-apiserver-proxy-testsrv1251	1/1
Running			
testsrv1251	kube-system	kube-controller-manager-testsrv1251	1/1
Running			
testsrv1251	kube-system	kube-proxy-4pp8v	1/1
Running			
testsrv1251	kube-system	kube-scheduler-testsrv1251	1/1
Running			
testsrv1251	kube-system	metrics-server-6f5fb84cdd-hkztz	1/1
Running			
testsrv1251	kube-system	tiller-deploy-84f4c8bb78-v6k95	1/1
Running			

- Optional: Define multiple Developer Portal endpoints. See [Defining multiple portal endpoints for a VMware environment](#).
- If you have now installed all subsystems, continue to [Post-deployment steps](#).

Note:

When you create or register a Developer Portal service, the Portal subsystem checks that the Portal web endpoint is accessible. However sometimes, for example due to the complexity of public and private networks, the endpoint cannot be reached. The following example shows the errors that you might see in the `portal-www` pod, admin container logs, if the endpoint cannot be reached:

```
An error occurred contacting the provided portal web endpoint: example.com
The provided Portal web endpoint example.com returned HTTP status code 504
```

In this instance, you can disable the Portal web endpoint check so that the Developer Portal service can be created successfully. To disable the endpoint check, complete the following update:

On VMware

In your `apicup` project, create a file called `ptl-extra-values.yaml` (or edit the file if one already exists), and add the following section:

```
spec:
  template:
    - containers:
      - env:
        - name: PORTAL_SKIP_WEB_ENDPOINT_VALIDATION
          value: "true"
        name: admin
      name: www
```

Run the following commands:

```
apicup subsys set <ptl_subsys> extra-values-file <path-to-ptl-extra-values-yaml-file>
apicup subsys install <ptl_subsys>
```

Configuring two NICs on the Developer Portal

Configure the Developer Portal to use two network interface cards.

About this task

When deploying the Developer Portal, you can optionally separate network traffic such that the portal has network interface in your private network, where the communications to the APIM manager take places, and another network interface in your public network, where the end users can access the Portal web sites.

You can do this by using the iface on `eth0` for your traffic and defining an extra iface definition (`eth1`) for your public interface. Do this as part of the initial configuration of the Developer Portal. Note the following requirements:

- iface `eth0` and `eth1` must have different subnets. This means that you cannot test the configuration by using two IP addresses on the same subnet, due to the routing table produced by the OVA at bootup.
- The hostname of the machine must be reachable via `eth0` and you must use `eth0` as the traffic (private interface) and `eth1` as the public interface.

Procedure

- Verify that you have created a first interface, by following the instructions in [Step 13 of Configuring the Developer Portal subsystem](#).
- Use `apicup` to add an extra iface definition as follows for your public interface:

```
apicup iface create test-stack-portal test0094.test.example.com eth1 1.2.3.232/255.255.254.0 1.2.3.1
```

- Set the hostname of your `portal-www` ingress to the hostname that is associated with the IP address given on `eth1`.

```
apicup subsys set my-portal portal-www portal.myhost0232.company.com
```

- When configuring the second NIC during initial installation, return to [Step 14 of Configuring the Developer Portal subsystem](#). Follow the remaining steps in the configuration instructions, and then create the Portal OVA and deploy it as usual.

Notes:

- If you are creating multiple network interfaces, each one must be on a different subnet with a different gateway.

- Before powering on the Portal, use the VMware GUI to add a second network card to the VM, with the correct settings for your public network.

Specifying a range of allowable client IP addresses for Developer Portal

Specify the range of client IP addresses to be allowed in a deployment of the Developer Portal

About this task

Use an extra-values-file, which serves as a portal CR override, to provide an allowable addresses annotation.

Procedure

In your `apicup` project, create a file called `pt1-extra-values.yaml` with the necessary content. For example:

```
spec:
  portalUIEndpoint:
    annotations:
      ingress.kubernetes.io/whitelist-source-range: 1.2.88.0/24
```

In this example, `1.2.88.0/24` is the acceptable range of client IP addresses.

Note: Depending on your network configuration for the forwarding of client IP addresses, you might need to run several test iterations to work out which client IP addresses to include in the list.

Defining multiple portal endpoints for a VMware environment

Multiple public facing endpoints (`portal-www`) can be defined for the Developer Portal.

About this task

You can override the single endpoint definition for `portal-www`, and the associated `portal-www-ingress` TLS certificate, to support multiple `portal-www` endpoints.

For information about the endpoints for the Developer Portal, see [Configuring the Developer Portal subsystem](#).

The task requires accessing your portal virtual appliance and creating Kubernetes secrets. Two methods are available for creating Kubernetes secrets:

- The recommended method uses a tool that exists on the portal appliance VM called `cert-manager`. To use the `cert-manager` method, skip step [5](#).
- If you do not want to use the `cert-manager` method, you can create your Kubernetes secrets manually. Follow step [5](#) instead of steps [3](#) and [4](#).

Procedure

Create `portal-www` endpoint certificates with `cert-manager`.

1. Log in to your portal virtual appliance with an SSH client, as `apicadm` user

```
ssh apicadm@<subsystem hostname>
```

2. Switch to the root user: `sudo -i`
3. Create a file called `ingress-issuer.yaml` and paste in the following content:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ingress-ca
spec:
  secretName: ingress-ca
  commonName: "ingress-ca"
  usages:
  - digital signature
  - key encipherment
  - cert sign
  isCA: true
  duration: 87600h # 10 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  issuerRef:
    name: selfsigning-issuer
    kind: Issuer
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
spec:
  ca:
    secretName: ingress-ca
```

Run `kubectl apply -f ingress-issuer.yaml`.

4. Create cert-manager certificate objects for each `portal-www` endpoint.

- a. Create a file called `portal-web-certs.yaml`, and paste in a `Certificate` entry for each `portal-www` endpoint. In this example, two endpoints are created, so the example file contains two certificates:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  labels:
    app.kubernetes.io/instance: portal
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: web-endpoint
  name: portal-web-host1
  namespace: default
spec:
  dnsNames:
  - <FQDN of portal web endpoint 1>
  duration: 17520h0m0s
  issuerRef:
    group: cert-manager.io
    kind: Issuer
    name: ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: portal-web-host1
  usages:
  - digital signature
  - key encipherment
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  labels:
    app.kubernetes.io/instance: portal
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: web-endpoint
  name: portal-web-host2
  namespace: default
spec:
  dnsNames:
  - <FQDN of portal web endpoint 2>
  duration: 17520h0m0s
  issuerRef:
    group: cert-manager.io
    kind: Issuer
    name: ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: portal-web-host2
  usages:
  - digital signature
  - key encipherment
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  labels:
    app.kubernetes.io/instance: portal
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: web-endpoint
  name: portal-web-host3
  namespace: default
spec:
  dnsNames:
  - <FQDN of portal web endpoint 3>
  duration: 17520h0m0s
  issuerRef:
    group: cert-manager.io
    kind: Issuer
    name: ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: portal-web-host3
  usages:
  - digital signature
  - key encipherment
```

- b. Apply the file:

```
kubectl apply -f portal-web-certs.yaml
```

Create `portal-www` without cert-manager.

5. Create Kubernetes secrets with OpenSSL. Follow this step instead of steps [3](#) and [4](#).

To create the secrets and certificates for each `portal-www` endpoint, complete the following steps:

- a. If you are using self-signed certificates, create the TLS secrets for each `portal-www` endpoint by generating the certificates. The following example shows how to generate certificates for each `portal-www` endpoint by using OpenSSL:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout pt1.host1-tls.key -out pt1.host1-tls.crt -subj
"/CN=pt1.host1.example.com"
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout pt1.host2-tls.key -out pt1.host2-tls.crt -subj
```

```
"/CN=ptl.host2.example.com"
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ptl.host3-tls.key -out ptl.host3-tls.crt -subj
"/CN=ptl.host3.example.com"
```

- b. Store the SSL certificates in a secret:
 - i. Copy the certificates to the Portal virtual appliance, for example:

```
scp ptl.host1-tls.key ptl.host1-tls.crt ptl.host2-tls.key ptl.host2-tls.crt ptl.host3-tls.key ptl.host3-tls.crt
apicadm@portal-vm-address
```

- ii. Log in to your portal virtual appliance with an SSH client, as `apicadm` user
- iii. Switch to the root user: `sudo -i`
- iv. Create the Kubernetes secrets:

```
kubectl --kubeconfig /etc/kubernetes/admin.conf create secret tls portal-web-host1 --key ptl.host1-tls.key --
cert ptl.host1-tls.crt
kubectl --kubeconfig /etc/kubernetes/admin.conf create secret tls portal-web-host2 --key ptl.host2-tls.key --
cert ptl.host2-tls.crt
kubectl --kubeconfig /etc/kubernetes/admin.conf create secret tls portal-web-host3 --key ptl.host3-tls.key --
cert ptl.host3-tls.crt
```

Add `portal-wwv` endpoints and secrets to your `ptl-extra-values.yaml` file.

6. Exit from the portal VM, and in your `apicup` project directory, create a file called `ptl-extra-values.yaml` with the content:

```
spec:
  portalUIEndpoint:
    annotations: # Remove if cert-manager not used
      cert-manager.io/issuer: ingress-issuer # Remove if cert-manager not used
    hosts:
      - name: ptl.host1.example.com
        secretName: portal-web-host1
      - name: ptl.host2.example.com
        secretName: portal-web-host2
      - name: ptl.host3.example.com
        secretName: portal-web-host3
```

Note: If you followed step 5 and did not use cert-manager to create your secrets, then delete the `annotations` section, so your `ptl-extra-values.yaml` contains:

```
spec:
  portalUIEndpoint:
    hosts:
      - name: ptl.host1.example.com
        secretName: portal-web-host1
      - name: ptl.host2.example.com
        secretName: portal-web-host2
      - name: ptl.host3.example.com
        secretName: portal-web-host3
```

Run the commands:

```
apicup subsys set <ptl_subsys> extra-values-file <path-to-ptl-extra-values-yaml-file>
apicup subsys install <ptl_subsys>
```

7. SSH back into your portal virtual appliance, and confirm that multiple hosts exist for your `ptl-portal-web` ingress:

```
sudo kubectl get ingress
NAME                                HOSTS                                                                 ADDRESS      PORTS      AGE
ptl-portal-director                 api.ptl.example.com                                                  80, 443     25m
ptl-portal-web                       ptl.host1.example.com,ptl.host2.example.com,ptl.host3.example.com  80, 443     25m
```

Deploying DataPower Gateway virtual appliance

API Connect uses IBM® DataPower® Gateway to provide the gateway service.

The instructions in this section describe how to deploy IBM DataPower Gateway from files obtained from [Passport Advantage®](#), and apply to the following API Connect scenarios:

- API Connect deployments for the VMware environment
- API Connect deployments for Kubernetes, with a DataPower Gateway in a *non-Kubernetes* environment.

For these scenarios, continue with [Installing DataPower Gateway](#).

Note: Do not use these instructions if you are deploying API Connect deployments for Kubernetes, with DataPower Gateway in a *Kubernetes* environment. In this case, you can use DataPower Gateway files that are packaged with API Connect, on the API Connect section of Fix Central, and configured by using the API Connect `apicup` commands. Follow the instructions in [Obtaining product files](#).

- [Installing DataPower Gateway](#)
Install and configure IBM DataPower Gateway in a non-Kubernetes environment for use with API Connect.
- [Configuring DataPower Gateway for API Connect](#)
You can configure the IBM DataPower Gateway to prepare for a registration with the API Connect Management server.
- [Sample configuration for multiple peering objects on gateway services external to Kubernetes](#)
Sample for reconfiguring the API Connect domain configuration for your gateway services to include multiple peering objects.

Installing DataPower Gateway

Install and configure IBM® DataPower® Gateway in a non-Kubernetes environment for use with API Connect.

Before you begin

Do not use these instructions for either of the following deployment scenarios:

- Both API Connect and DataPower Gateway in a Kubernetes environment.
In this scenario, use the instructions in [Obtaining product files](#).
- An existing DataPower Gateway in a non-Kubernetes environment, and you want to install a newer Fix Pack.
In this scenario, complete an *upgrade* rather than a new installation. See [Upgrading DataPower Gateway Service](#).

To review the installation and configuration scenarios, see [Deploying DataPower Gateway virtual appliance](#).

Procedure

1. Ensure that DataPower Gateway firmware version you plan to install is compatible with the API Connect Management server version.

Note:

You can use any combination of API Connect 10.0.5.x with DataPower Gateway 10.5.0.x or DataPower API Gateway 10.5.0.x.

Before you install, best practice is to review the latest compatibility support for your version of API Connect. To view compatibility support, follow the instructions in [IBM API Connect Version 10 software product compatibility requirements](#) to access API Connect information on the Software Product Compatibility Reports website. Once you access information for your version of API Connect, select Supported Software_> Integration Middleware, and view the list of compatible DataPower Gateway versions.

2. Obtain your DataPower files from IBM [Passport Advantage](#).
Note that the DataPower files from [Passport Advantage](#) have entitled services for API Connect users.

See also [Download the firmware image from Passport Advantage](#).
3. Follow the DataPower Gateway installation information: <https://www.ibm.com/docs/en/datapower-gateway>.
Note: Set the timezone to UTC on all DataPower installations for use with API Connect.
4. Continue with [Configuring DataPower Gateway for API Connect](#).

Configuring DataPower Gateway for API Connect

You can configure the IBM® DataPower® Gateway to prepare for a registration with the API Connect Management server.

About this task

API Connect and DataPower supports two different types of gateway configurations. The DataPower Gateway (v5 compatible) provides the same support as the gateway support that was available with API Connect version 5.x. The DataPower API Gateway is an enhanced gateway that is performance-focused. See [API Connect gateway types](#) for more information about the differences between the gateway types.

Continue with the instructions for the type of Gateway Service you are configuring:

- [Configuring DataPower API Gateway](#)
Configure the DataPower API Gateway to prepare for registration with the API Connect Management server.
- [Configuring DataPower Gateway \(v5 compatible\)](#)
You can configure the DataPower Gateway (v5 compatible) to prepare for a registration with the API Connect Management server.

Configuring DataPower API Gateway

Configure the DataPower® API Gateway to prepare for registration with the API Connect Management server.

Before you begin

- These instructions are for DataPower API Gateway deployments in a non-Kubernetes environment. Do not use these instructions if you installed API Connect on Kubernetes, with your DataPower API Gateway Service also in the Kubernetes environment. To review deployment scenarios, see [Deploying DataPower Gateway virtual appliance](#).
- Ensure you have installed a version of DataPower API Gateway that is compatible with the version of the API Connect Management server. See [Installing DataPower Gateway](#).
- A shared certificate and private key is used for securing the communication between the API Connect Management server and the gateway. See *Generating keys and certificates* in the appropriate version of the [DataPower documentation](#) for instructions on how to create them with the DataPower tools.
- Ensure that the time zone for the DataPower API Gateway is set to UTC.

About this task

- These instructions provide the basic steps for configuring a gateway service with a single gateway server. The lowest-level configuration objects are created first, then used in other configuration objects.

- Adding gateways to configure a peering environment is similar to creating the first gateway, and is recommended for resiliency in a production environment. A minimum of three gateway servers in a gateway service is recommended for high availability. See *Gateway peering* in the [DataPower documentation](#) for more information about configuring additional gateways for peering. See *Providing the gateway service to API Connect* in the [DataPower documentation](#) for more details about the DataPower settings and procedures.

Procedure

To configure a DataPower API Gateway to communicate with API Connect, complete the following steps:

Note: Use these instructions only for DataPower API Gateway . If you are configuring DataPower Gateway (v5 compatible) see [Configuring DataPower Gateway \(v5 compatible\)](#).

1. Open the DataPower WebGUI interface.
Most of the configuration procedure is done in the DataPower WebGUI interface, not in the Blueprint Console.
2. Optional: Enable the XML management interface in the default domain.
The XML management interface is optional for DataPower API Gateway. If enabled, this interface allows you to send status and configuration requests to the DataPower appliance through a standard SOAP interface, using SOAP messages.
 - a. Search for XML management interface in the navigation search bar, and select it.
 - b. Set the Administrative state to enabled.
 - c. You can specify a different port number if you do not want to use the default of **5550**.
 - d. Select Apply to make the changes
 - e. Save changes to the default domain by selecting Save Configuration.
3. Optional: Enable the REST management interface in the default domain.
The REST management interface is required if you want to enable to the Trace feature in the assembly Test tab in the API Manager.
 - a. Search for REST management interface in the navigation search bar, and select it.
 - b. Set the Administrative state to enabled.
 - c. You can specify a different port number if you do not want to use the default of **5554**.
 - d. Select Apply to make the changes
 - e. Save changes to the default domain by selecting Save Configuration.
4. Create an application domain.
This domain receives your traffic. The name of the DataPower domain where you configure the API Connect Gateway Service must be the same on each DataPower Gateway.
 - a. Search for Application domain in the navigation search bar, and select it.
 - b. Select Add to create the application domain.
 - c. Enter a unique name for your domain.
 - d. Ensure that enabled is selected for the Administrative state.
 - e. Ensure that the default domain is listed in the Visible application domain list.
 - f. Select Apply.
 - g. Change to your new application domain by selecting Domain in the menu bar, and selecting the domain that you created.
 - h. Select Save changes and switch domains.
All of the remaining steps on the DataPower gateway must be done in the application domain that you created.
 - i. Save changes to the domain by selecting Save Configuration.
5. Ensure that your deployment includes an NTP server to synchronize time between each of the DataPower Gateways.
See *Managing the NTP service* in the [DataPower documentation](#).
6. Ensure that you have set a unique **System Identifier** for each v10 DataPower gateway. See *Initializing the DataPower Gateway* in the [DataPower documentation](#).
7. Create a self-signed certificate and private key to be used to protect the traffic between the management server and the API gateway service process. You can generate a certificate and private key using DataPower or by using other tools, such as **OpenSSL**. See *Generating keys and certificates* in the appropriate version of the [DataPower documentation](#) for instructions on how to create a crypto key with the DataPower tools.
8. Upload your private crypto key file to the domain.
 - a. Search for Crypto key in the navigation search bar, and select it.
 - b. Select Add to create a key object.
 - c. Create a unique name for the key object in the Name field.
 - d. Select Upload...
 - e. Browse for the key file (which must be a .pem or .p12 file) and select it.
 - f. If you want to rename it, enter a new name for the file.
 - g. Select Upload to move it to the server in the cert:// folder.
 - h. Select Apply to save the changes.
9. Upload your crypto certificate file to the domain.
Note: If your certificate is signed by an Intermediate CA, you must include the entire chain in a single key file (either .pem or .p12) for uploading.
 - a. Search for Crypto certificate in the navigation search bar, and select it.
 - b. Select Add to create a certificate object.
 - c. Create a unique name for the certificate object in the Name field.
 - d. Select Upload...
 - e. Browse for the key file (which must be a .pem or .p12 file) and select it.
 - f. If you want to rename it, enter a new name for the file.
 - g. Select Upload to move it to the server in the cert:// folder.
 - h. Select Apply to save the changes.
10. Associate the Crypto key with the Crypto certificate by setting the Identification credential.
 - a. Search for Crypto Identification Credentials in the navigation search bar, and select it.
 - b. Select Add.
 - c. Enter a name for your credential.
 - d. Ensure that the Administrative state has a value of enabled.
 - e. In the Crypto Key field, select the name of the key object that you created from the drop-down menu.
 - f. In the Certificate object field, select the name of the certificate object that you created from the drop-down menu.
 - g. Select Apply to commit your changes.
11. Create your TLS Client profile.
 - a. Search for TLS Client profile in the navigation search bar, and select it.

- b. Select Add to create a client profile.
 - c. Create a unique name for the profile in the Name field.
 - d. Select your Identification credential from the drop-down list.
 - e. Ensure that the value of Validate server certificate is set to off.
 - f. Ensure that the value of Use SNI is set to on.
 - g. Select Apply to save the changes.
12. Create your TLS Server profile.
- a. Search for TLS Server Profile in the navigation search bar, and select it.
 - b. Select Add to create a server profile.
 - c. Create a unique name for the profile in the Name field.
 - d. Select your Identification credential from the drop-down list.
 - e. Ensure that the value of Request client authentication is set to off.
Disabling the request for client authentication for the TLS client profile and the validation of server certificates for the TLS server profile disables security for the ease of configuring the gateway. To enable the secure communication using mutual TLS between IBM API Connect and Gateway, see [Binding a TLS server profile to a gateway service](#).
- If mTLS is disabled it is recommended to use JWT application layer security instead. See [Enable JWT instead of mTLS](#).
- f. Select Apply to save the changes.
13. Define a configuration sequence.
- The configuration sequence specifies how DataPower implements the APIs that are defined in API Connect.
- a. Search for Configuration sequence in the navigation search bar, and select it.
 - b. Select Add.
 - c. Enter a name for your configuration sequence.
The name `api-c-config` is not allowed because it is already used internally.
 - d. Ensure that the Administrative state has a value of enabled.
 - e. Ensure that the value in the Location profiles field is set to local:///.
This is the default value, so you might not need to change it.
 - f. Select the Access profile.
For instructions on creating access profiles, see *Configuring the access profile for a configuration sequence* in the appropriate version of the [DataPower documentation](#).
 - g. Change the value of the Configuration execution interval field to 3000.
The other fields can retain their default settings.
 - h. Select Apply to commit your changes.
14. Configure your gateway peering object for the API Connect Gateway Service.
- This step is required when you set up a peer group of gateways, even if there is only a single gateway server in the gateway service.
- a. Search for Gateway peering in the navigation search bar, and select it.
 - b. Select Add.
 - c. Enter a unique name for your gateway peering object.
 - d. Ensure that the Administrative state has a value of enabled.
 - e. Select a local address for the communications among the members of the peer group.
 - f. Select a local port for the communication.
You can use the default value of 16380.
 - g. Select a monitor port for the communication.
You can use the default value of 26380.
 - h. Because this procedure uses only one gateway, ensure that Peer group mode is not selected.
 - i. Clear the Enable TLS check box. TLS is not needed for a single peer.
 - j. Set the Persistence location value to **Memory** for either physical DataPower appliance or virtual DataPower appliance.
 - k. Select Apply to commit your changes.
15. Configure your gateway peering object for rate limit information.
- a. Search for Gateway peering in the navigation search bar, and select it.
 - b. Select Add.
 - c. Enter a unique name for your gateway peering object.
 - d. Ensure that the Administrative state has a value of enabled.
 - e. Select a local address for the communications among the members of the peer group.
 - f. Select a local port for the communication.
Use a unique port, different than the ports used for communication by other gateway peering objects.
 - g. Select a monitor port for the communication.
Use a unique port, different than the ports used for monitoring by other gateway peering objects.
 - h. Because this procedure uses only one gateway, ensure that Peer group mode is not selected.
 - i. Clear the Enable TLS check box. TLS is not needed for a single peer.
 - j. Set the Persistence location value to **Memory** for either physical DataPower appliance or virtual DataPower appliance.
 - k. Select Apply to commit your changes.
16. Optional: If you are using the ratelimit module described in the [DataPower documentation](#), configure your gateway peering object for GatewayScript rate limiting information.
- a. Search for Gateway peering in the navigation search bar, and select it.
 - b. Select Add.
 - c. Enter a unique name for your gateway peering object.
 - d. Ensure that the Administrative state has a value of enabled.
 - e. Select a local address for the communications among the members of the peer group.
 - f. Select a local port for the communication.
Use a unique port, different than the ports used for communication by other gateway peering objects.
 - g. Select a monitor port for the communication.
Use a unique port, different than the ports used for monitoring by other gateway peering objects.
 - h. Because this procedure uses only one gateway, ensure that Peer group mode is not selected.
 - i. Clear the Enable TLS check box. TLS is not needed for a single peer.
 - j. Set the Persistence location value to **Memory** for either physical DataPower appliance or virtual DataPower appliance.
 - k. Select Apply to commit your changes.

17. Configure your gateway peering object for subscription information.
 - a. Search for Gateway peering in the navigation search bar, and select it.
 - b. Select Add.
 - c. Enter a unique name for your gateway peering object.
 - d. Ensure that the Administrative state has a value of enabled.
 - e. Select a local address for the communications among the members of the peer group.
 - f. Select a local port for the communication.

Use a unique port, different than the ports used for communication by other gateway peering objects.
 - g. Select a monitor port for the communication.

Use a unique port, different than the ports used for monitoring by other gateway peering objects.
 - h. Because this procedure uses only one gateway, ensure that Peer group mode is not selected.
 - i. Clear the Enable TLS check box. TLS is not needed for a single peer.
 - j. Set the Persistence location value to **Memory** for either physical DataPower appliance or virtual DataPower appliance.
 - k. Select Apply to commit your changes.
18. Configure the gateway peering object for the API probe.

In order for API Connect to receive trace data in the Test tab's debugger, the DataPower API Gateway must be configured to support the API probe.

 - a. Search for Gateway peering in the navigation search bar, and select it.
 - b. Select Add.
 - c. Enter a unique name for your gateway peering object.
 - d. Ensure that the Administrative state has a value of enabled.
 - e. Select a local address for the communications among the members of the peer group.
 - f. Select a local port for the communication.

Use a unique port, different than the ports used for communication by other gateway peering objects.
 - g. Select a monitor port for the communication.

Use a unique port, different than the ports used for monitoring by other gateway peering objects.
 - h. Because this procedure uses only one gateway, ensure that Peer group mode is not selected.
 - i. Clear the Enable TLS check box. TLS is not needed for a single peer.
 - j. Set the Persistence location value to **Memory** for either physical DataPower appliance or virtual DataPower appliance.
 - k. Select Apply to commit your changes.
19. Configure the gateway peering manager.
 - a. Search for Gateway Peering Manager in the navigation search bar, and select it.
 - b. Set the Administrative state to enabled.
 - c. In the pull-down menu next to API Connect Gateway Service, select the gateway peering object configured in Step 14 for the API Connect Gateway Service.
 - d. In the pull-down menu next to Rate Limit, select the gateway peering object configured in Step 15 for rate limit information.
 - e. In the pull-down menu next to Subscription, select the gateway peering object configured in Step 17 for subscription.
 - f. In the pull-down menu next to API Probe, select the gateway peering object that you configured in Step 18 for the API probe.
 - g. If you are using the ratelimit module described in the [DataPower documentation](#), then for Gateway rate limiting, select the gateway peering object that you configured in Step 16 for rate limit information.
 - h. Select Apply to commit your changes.
20. Configure the API probe settings object.
 - a. Search for API probe settings in the navigation search bar, and select it.
 - b. Set the Administrative state to enabled.
 - c. Set the maximum number of records to 1000.
 - d. Set the expiration to 60 minutes.
 - e. In the pull-down menu next to Gateway Peering, select the gateway peering object that you configured in Step 18 for the API probe.
 - f. Select Apply to commit your changes.
21. Set the API Connect Gateway service to define the communication interface with the API Connect Management server and for API transactions.
 - a. Search for API Connect Gateway service in the navigation search bar, and select it.
 - b. Ensure that the Administrative state is set to enabled.
 - c. In the Local address field, enter the IP address of the DataPower gateway to which you want the traffic from the API Connect Management server to be sent.
 - d. Specify a value for Local port. You can use the default value of 3000, or specify a different port value.

Note: The Local port specifies the port through which API Connect connects to manage the API Connect Gateway Service. Use this port when you configure a Gateway Service on API Connect. Beyond this port, the gateway service uses two additional consecutive ports after the defined local port to bind to a loopback address. Therefore, you must ensure that there are no conflicts on all three consecutive ports that start from the defined local port.
 - e. In the TLS client field drop-down list, select the name of the TLS client profile that you created.
 - f. In the TLS server field drop-down list, select the name of the TLS server profile that you created.
 - g. In the API gateway address field, enter the IP address for the DataPower gateway to which you want the API traffic sent.
 - h. Use the default port value of 9443 for the API gateway port.

If the port is not being used by another service, you can also change it to port 443 if you want API transactions to be sent to the default port for HTTPS.
 - i. For DataPower API Gateway, set the Gateway Peering to (**none**).

When no gateway peering object is configured for the DataPower API Gateway, the peering configuration defined in the Gateway Peering Manager configuration is used.
 - j. Select whether you want the DataPower Gateway (v5 compatible) or the DataPower API Gateway.

When the option is selected, it enables the registration of a DataPower Gateway (v5 compatible) gateway. Clear it to enable a DataPower API Gateway.
22. Optional: Enable JWT security instead of mTLS for communication from management to gateway. JWT security provides application layer security and can be used instead of mTLS when load-balancers are located between subsystems that require TLS termination. For more information about JWT security, see [Enable JWT instead of mTLS](#).

To enable JWT and disable mTLS on the gateway appliance, see *Configuring the API Connect gateway service* in the [DataPower documentation](#). The JWKS URL required to enable JWT security can be found in the management subsystem settings:

```

apicup subsys get <management subsystem>

...
jwks-url https://appliance1.apic.acme.com/api/cloud/oauth2/certs JWKS URL for Portal and Analytics subsystems to
validate JWT -- this is unsettingtable and is generated based on the platform-api endpoint
...

```

23. Register the gateway service in the API Connect Cloud Manager console:
 - a. Open the API Connect Cloud Manager console.
 - b. Navigate to Configure Topology.

- c. Select Register Service.
- d. Select DataPower API Gateway for the DataPower API Gateway.
- e. Add a title, name, and summary for the gateway connection.
- f. Optional: Configure the OAuth Shared Secret.
This setting allows OAuth tokens to be shared across multiple gateway services.
- g. Enter one of the following values in the API Invocation Endpoint field:
 - IP address of the load balancer for the API transactions
 - IP address or host name of one of the gateways
 For example: `https://192.0.2.0:9443/`
- h. Enter the one of the following values in the Management Endpoint field:
 - IP address of the load balancer for the management server traffic set to port 3000
 - IP address or hostname of one of the gateways
 For example: `https://192.0.2.0:3000/`
- 24. Select the default TLS Client Profile
- 25. Optional: Configure Server Name Indication (SNI) profiles.
SNI profiles allow different TLS certificates to be used for API transaction requests from different host names.

Configuring DataPower Gateway (v5 compatible)

You can configure the DataPower® Gateway (v5 compatible) to prepare for a registration with the API Connect Management server.

Before you begin

- These instructions are for DataPower API Gateway deployments in a non-Kubernetes environment. Do not use these instructions if you installed API Connect on Kubernetes, with your DataPower API Gateway Service also in the Kubernetes environment. To review deployment scenarios, see [Deploying DataPower Gateway virtual appliance](#).
- Ensure you have installed a version of DataPower API Gateway that is compatible with the version of the API Connect Management server. See [Installing DataPower Gateway](#).
- A shared certificate and private key is used for securing the communication between the API Connect Management server and the gateway. See *Generating keys and certificates* in the appropriate version of the [DataPower documentation](#) for instructions on how to create them with the DataPower tools.
- Ensure that the time zone for the DataPower API Gateway is set to UTC.

About this task

- These instructions provide the basic steps for configuring a gateway service with a single gateway server. The lowest-level configuration objects are created first, then used in other configuration objects.
- Adding gateways to configure a peering environment is similar to creating the first gateway, and is recommended for resiliency in a production environment. A minimum of three gateway servers in a gateway service is recommended for high availability. See *Gateway peering* in the [DataPower documentation](#) for more information about configuring additional gateways for peering. See *Providing the gateway service to API Connect* in the [DataPower documentation](#) for more details about the DataPower settings and procedures.

Procedure

To configure a DataPower gateway to communicate with API Connect, complete the following steps:

Note: Use these instructions only for DataPower Gateway (v5 compatible). If you are configuring DataPower API Gateway, see [Configuring DataPower API Gateway](#).

1. Open the DataPower WebGUI interface.
Most of the configuration procedure is done in the DataPower WebGUI interface, not in the Blueprint Console.
2. Enable the XML management interface in the **default** domain, if required. The XML management interface is required for DataPower Gateway (v5 compatible). The XML management interface is optional for DataPower API Gateway.
 - a. Search for **XML management interface** in the navigation search bar, and select it.
 - b. Set the Administrative state to enabled.
 - c. You can specify a different port number if you do not want to use the default of **5550**.
 - d. Select Apply to make the changes
 - e. Save changes to the default domain by selecting Save Configuration.
3. Create an application domain.
This domain receives your traffic. The name of the DataPower domain where you configure the API Connect Gateway Service must be the same on each DataPower Gateway.
 - a. Search for Application domain in the navigation search bar, and select it.
 - b. Select Add to create the application domain.
 - c. Enter a unique name for your domain.
 - d. Ensure that enabled is selected for the Administrative state.
 - e. Ensure that the default domain is listed in the Visible application domain list.
 - f. Select Apply.
 - g. Change to your new application domain by selecting Domain in the menu bar, and selecting the domain that you created.
 - h. Select Save changes and switch domains.
All of the remaining steps on the DataPower gateway must be done in the application domain that you created.
 - i. Save changes to the domain by selecting Save Configuration.
4. For the DataPower Gateway (v5 compatible) only: Enable statistics in the domain you created for **API Connect**.
 - a. Search for and select Statistics settings in the navigation search.
 - b. Select enabled for the Administrative state.
 - c. Select Apply.
5. Ensure that your deployment includes an NTP server to synchronize time between each of the DataPower Gateways.
See [Managing the NTP service](#).

6. Ensure that you have set a unique **System Identifier** for each v10 DataPower gateway. See [Initializing the DataPower Gateway](#).
7. Create a self-signed certificate and private key to be used to protect the traffic between the management server and the API gateway service process. You can generate a certificate and private key using DataPower or by using other tools, such as **OpenSSL**. See *Generating keys and certificates* in the appropriate version of the [DataPower documentation](#) for instructions on how to create a crypto key with the DataPower tools.
8. Upload your private crypto key file to the domain.
 - a. Search for Crypto key in the navigation search bar, and select it.
 - b. Select Add to create a key object.
 - c. Create a unique name for the key object in the Name field.
 - d. Select Upload....
 - e. Browse for the key file (which must be a .pem or .p12 file) and select it.
 - f. If you want to rename it, enter a new name for the file.
 - g. Select Upload to move it to the server in the cert:// folder.
 - h. Select Apply to save the changes.
9. Upload your crypto certificate file to the domain.

Note: If your certificate is signed by an Intermediate CA, you must include the entire chain in a single key file (either .pem or .p12) for uploading.

 - a. Search for Crypto certificate in the navigation search bar, and select it.
 - b. Select Add to create a certificate object.
 - c. Create a unique name for the certificate object in the Name field.
 - d. Select Upload....
 - e. Browse for the key file (which must be a .pem or .p12 file) and select it.
 - f. If you want to rename it, enter a new name for the file.
 - g. Select Upload to move it to the server in the cert:// folder.
 - h. Select Apply to save the changes.
10. Associate the Crypto key with the Crypto certificate by setting the Identification credential.
 - a. Search for Crypto Identification Credentials in the navigation search bar, and select it.
 - b. Select Add.
 - c. Enter a name for your credential.
 - d. Ensure that the Administrative state has a value of enabled.
 - e. In the Crypto Key field, select the name of the key object that you created from the drop-down menu.
 - f. In the Certificate object field, select the name of the certificate object that you created from the drop-down menu.
 - g. Select Apply to commit your changes.
11. Create your TLS Client profile.
 - a. Search for TLS Client profile in the navigation search bar, and select it.
 - b. Select Add to create a client profile.
 - c. Create a unique name for the profile in the Name field.
 - d. Select your Identification credential from the drop-down list.
 - e. Ensure that the value of Validate server certificate is set to off.
 - f. Ensure that the value of Use SNI is set to on.
 - g. Select Apply to save the changes.
12. Create your TLS Server profile.
 - a. Search for TLS Server Profile in the navigation search bar, and select it.
 - b. Select Add to create a server profile.
 - c. Create a unique name for the profile in the Name field.
 - d. Select your Identification credential from the drop-down list.
 - e. Ensure that the value of Request client authentication is set to off.

Disabling the request for client authentication for the TLS client profile and the validation of server certificates for the TLS server profile disables security for the ease of configuring the gateway. To enable the secure communication using mutual TLS between IBM API Connect and Gateway, see [Binding a TLS server profile to a gateway service](#).

If mTLS is disabled it is recommended to use JWT application layer security instead. See [Enable JWT instead of mTLS](#).
 - f. Select Apply to save the changes.
13. Configure your gateway peering object for the API Connect Gateway Service.

This step is required when you set up a peer group of gateways, even if there is only a single gateway server in the gateway service.

 - a. Search for Gateway peering in the navigation search bar, and select it.
 - b. Select Add.
 - c. Enter a unique name for your gateway peering object.
 - d. Ensure that the Administrative state has a value of enabled.
 - e. Select a local address for the communications among the members of the peer group.
 - f. Select a local port for the communication.

You can use the default value of 16380.
 - g. Select a monitor port for the communication.

You can use the default value of 26380.
 - h. Because this procedure uses only one gateway, ensure that Peer group mode is not selected.
 - i. Clear the Enable TLS check box. TLS is not needed for a single peer.
 - j. Set the Persistence location value to **Memory** for either physical DataPower appliance or virtual DataPower appliance.
 - k. Select Apply to commit your changes.
14. Set the API Connect Gateway service to define the communication interface with the API Connect Management server and for API transactions.
 - a. Search for API Connect Gateway service in the navigation search bar, and select it.
 - b. Ensure that the Administrative state is set to enabled.
 - c. In the Local address field, enter the IP address of the DataPower gateway to which you want the traffic from the API Connect Management server to be sent.
 - d. Use the default port value of 3000 for the Local port.
 - e. In the TLS client field drop-down list, select the name of the TLS client profile that you created.
 - f. In the TLS server field drop-down list, select the name of the TLS server profile that you created.
 - g. In the API gateway address field, enter the IP address for the DataPower gateway to which you want the API traffic sent.
 - h. Use the default port value of 9443 for the API gateway port.

If the port is not being used by another service, you can also change it to port 443 if you want API transactions to be sent to the default port for HTTPS.
 - i. For DataPower Gateway (v5 compatible), select the gateway peering object that you created in Step [13](#).
 - j. Select whether you want the DataPower Gateway (v5 compatible) or the DataPower API Gateway.

When the option is selected, it enables the registration of a DataPower Gateway (v5 compatible) gateway.

15. Optional: Enable JWT security instead of mTLS for communication from management to gateway. JWT security provides application layer security and can be used instead of mTLS when load-balancers are located between subsystems that require TLS termination. For more information about JWT security, see [Enable JWT instead of mTLS](#).

To enable JWT and disable mTLS on the gateway appliance, see *Configuring the API Connect gateway service* in the [DataPower documentation](#). The JWKS URL required to enable JWT security can be found in the management subsystem settings:

```
apicup subsys get <management subsystem>

...
jwks-url https://appliance1.apic.acme.com/api/cloud/oauth2/certs JWKS URL for Portal and Analytics subsystems to
validate JWT -- this is unsettable and is generated based on the platform-api endpoint
...
```

16. Register the gateway service in the API Connect Cloud Manager console:
- Open the API Connect Cloud Manager console.
 - Navigate to Configure Topology.
 - Select Register Service.
 - Select DataPower Gateway (v5 compatible) for the gateway that was available in version 5.
 - Add a title, name, and summary for the gateway connection.
 - Optional: Configure the OAuth Shared Secret.
This setting allows OAuth tokens to be shared across multiple gateway services.
 - Enter one of the following values in the API Invocation Endpoint field:
 - IP address of the load balancer for the API transactions
 - IP address or host name of one of the gatewaysFor example: `https://192.0.2.0:9443/`
 - Enter the one of the following values in the Management Endpoint field:
 - IP address of the load balancer for the management server traffic set to port 3000
 - IP address or hostname of one of the gatewaysFor example: `https://192.0.2.0:3000/`
17. Select the default TLS Client Profile
18. Optional: Configure Server Name Indication (SNI) profiles.
SNI profiles allow different TLS certificates to be used for API transaction requests from different host names.

Sample configuration for multiple peering objects on gateway services external to Kubernetes

Sample for reconfiguring the API Connect domain configuration for your gateway services to include multiple peering objects.

- This reference shows how to configure API Connect domain configuration with gateway-peering configuration. To review domain configuration settings, see [Configuring DataPower Gateway for API Connect](#).
- This sample applies only to gateway services that are deployed external to Kubernetes, on either a physical or virtual appliance. This sample does not apply to gateway services that are deployed on Kubernetes.

The following examples show sample configuration for 3 different gateways, listed here with sample IP addresses of 1.1.1.1, 2.2.2.2, and 3.3.3.3. Note the following:

- The priority should be set differently on each of the three gateways. Set the lowest priority for the gateway that will run as primary.
- persistence** should be set to **memory** for all configured peering objects.
- Optionally, add SSL configuration.

Configuration on gateway 1 (1.1.1.1)

```
top; configure terminal;
domain apiconnect; visible default; exit;

sw apiconnect;

loglevel debug;
logging target gwd-log
type file
format text
timestamp syslog
size 50000
local-file logtemp:///gwd-log
event apic-gw-service debug
exit

config-sequence "apiconnect"
location "local:://"
watch "on"
delete-unused "on"
match "(.*)\.cfg$"
summary "Toolkit Reboot configuration"
run-sequence-interval 3000
optimize-for-apic on
exit

crypto
key sth_apic sharedcert:///sth_apic-privkey.cer
certificate sth_apic sharedcert:///sth_apic-sscert.cer
idcred sth_apic sth_apic sth_apic
ssl-client gwd_to_mgmt
idcred sth_apic
no validate-server-cert
exit
```

```
ssl-server gwd_to_mgmt
idcred sth_apic
no request-client-auth
validate-client-cert off
exit
exit
```

```
gateway-peering subs
admin enabled
local-address 1.1.1.1
local-port 15222
monitor-port 26222
priority 100
enable-ssl off
enable-peer-group on
peer 2.2.2.2
peer 3.3.3.3
persistence memory
exit
```

```
gateway-peering rate-limit
admin enabled
local-address 1.1.1.1
local-port 15223
monitor-port 26223
priority 100
enable-ssl off
enable-peer-group on
peer 2.2.2.2
peer 3.3.3.3
persistence memory
exit
```

```
gateway-peering gwd
admin enabled
local-address 1.1.1.1
local-port 15224
monitor-port 26224
priority 100
enable-ssl off
enable-peer-group on
peer 2.2.2.2
peer 3.3.3.3
persistence memory
exit
```

```
gateway-peering probe
admin enabled
local-address 1.1.1.1
local-port 15225
monitor-port 26225
priority 100
enable-ssl off
enable-peer-group on
peer 2.2.2.2
peer 3.3.3.3
persistence memory
exit
```

```
gateway-peering gws-rate-limit
admin enabled
local-address 1.1.1.1
local-port 15226
monitor-port 26226
priority 100
enable-ssl off
enable-peer-group on
peer 2.2.2.2
peer 3.3.3.3
persistence memory
exit
```

```
gateway-peering-manager
admin enabled
apic-gw-service gwd
rate-limit rate-limit
subscription subs
apiprobe probe
ratelimit-module gws-rate-limit
```

```
apic-gw-service
admin-state enabled
local-address 0.0.0.0
local-port 3000
api-gw-address 0.0.0.0
api-gw-port 9443
v5-compatibility-mode off
ssl-server gwd_to_mgmt
ssl-client gwd_to_mgmt
exit
```

```
write mem
```

Configuration on gateway 2 (2.2.2.2)

```

top; configure terminal;
domain apiconnect; visible default; exit;

sw apiconnect;

loglevel debug;
logging target gwd-log
type file
format text
timestamp syslog
size 50000
local-file logtemp:///gwd-log
event apic-gw-service debug
exit

config-sequence "apiconnect"
location "local:/// "
watch "on"
delete-unused "on"
match "(.*)\.cfg$"
summary "Toolkit Reboot configuration"
run-sequence-interval 3000
optimize-for-apic on
exit

crypto
key sth_apic sharedcert:///sth_apic-privkey.cer
certificate sth_apic sharedcert:///sth_apic-sscert.cer
idcred sth_apic sth_apic sth_apic
ssl-client gwd_to_mgmt
idcred sth_apic
no validate-server-cert
exit
ssl-server gwd_to_mgmt
idcred sth_apic
no request-client-auth
validate-client-cert off
exit
exit

gateway-peering subs
admin enabled
local-address 2.2.2.2
local-port 15222
monitor-port 26222
priority 105
enable-ssl off
enable-peer-group on
peer 1.1.1.1
peer 3.3.3.3
persistence memory
exit

gateway-peering rate-limit
admin enabled
local-address 2.2.2.2
local-port 15223
monitor-port 26223
priority 105
enable-ssl off
enable-peer-group on
peer 1.1.1.1
peer 3.3.3.3
persistence memory
exit

gateway-peering gwd
admin enabled
local-address 2.2.2.2
local-port 15224
monitor-port 26224
priority 105
enable-ssl off
enable-peer-group on
peer 1.1.1.1
peer 3.3.3.3
persistence memory
exit

gateway-peering probe
admin enabled
local-address 2.2.2.2
local-port 15225
monitor-port 26225
priority 105
enable-ssl off
enable-peer-group on
peer 1.1.1.1
peer 3.3.3.3
persistence memory
exit

gateway-peering gws-rate-limit
admin enabled
local-address 2.2.2.2
local-port 15226

```

```

monitor-port 26226
priority 105
enable-ssl off
enable-peer-group on
peer 1.1.1.1
peer 3.3.3.3
persistence memory
exit

gateway-peering-manager
admin enabled
apic-gw-service gwd
rate-limit rate-limit
subscription subs
apiprobe probe
ratelimit-module gws-rate-limit
exit

apic-gw-service
admin-state enabled
local-address 0.0.0.0
local-port 3000
api-gw-address 0.0.0.0
api-gw-port 9443
v5-compatibility-mode off
ssl-server gwd_to_mgmt
ssl-client gwd_to_mgmt
exit

write mem

```

Configuration on gateway 3 (3.3.3.3)

```

top; configure terminal;
domain apiconnect; visible default; exit;

sw apiconnect;

loglevel debug;
logging target gwd-log
type file
format text
timestamp syslog
size 50000
local-file logtemp:///gwd-log
event apic-gw-service debug
exit

config-sequence "apiconnect"
location "local:/// "
watch "on"
delete-unused "on"
match "(.*)\.cfg$"
summary "Toolkit Reboot configuration"
run-sequence-interval 3000
optimize-for-apic on
exit

crypto
key sth_apic sharedcert:///sth_apic-privkey.cer
certificate sth_apic sharedcert:///sth_apic-sscrt.cer
idcred sth_apic sth_apic sth_apic
ssl-client gwd_to_mgmt
idcred sth_apic
no validate-server-cert
exit
ssl-server gwd_to_mgmt
idcred sth_apic
no request-client-auth
validate-client-cert off
exit
exit

gateway-peering subs
admin enabled
local-address 3.3.3.3
local-port 15222
monitor-port 26222
priority 110
enable-ssl off
enable-peer-group on
peer 1.1.1.1
peer 2.2.2.2
persistence memory
exit

gateway-peering rate-limit
admin enabled
local-address 3.3.3.3
local-port 15223
monitor-port 26223
priority 110
enable-ssl off
enable-peer-group on
peer 1.1.1.1
peer 2.2.2.2

```

```

persistence memory
exit

gateway-peering gwd
admin enabled
local-address 3.3.3.3
local-port 15224
monitor-port 26224
priority 110
enable-ssl off
enable-peer-group on
peer 1.1.1.1
peer 2.2.2.2
persistence memory
exit

gateway-peering probe
admin enabled
local-address 3.3.3.3
local-port 15225
monitor-port 26225
priority 110
enable-ssl off
enable-peer-group on
peer 1.1.1.1
peer 2.2.2.2
persistence memory
exit

gateway-peering gws-rate-limit
admin enabled
local-address 3.3.3.3
local-port 15226
monitor-port 26226
priority 110
enable-ssl off
enable-peer-group on
peer 1.1.1.1
peer 2.2.2.2
persistence memory
exit

gateway-peering-manager
admin enabled
apic-gw-service gwd
rate-limit rate-limit
subscription subs
apiprobe probe
ratelimit-module gws-rate-limit
exit

apic-gw-service
admin-state enabled
local-address 0.0.0.0
local-port 3000
api-gw-address 0.0.0.0
api-gw-port 9443
v5-compatibility-mode off
ssl-server gwd_to_mgmt
ssl-client gwd_to_mgmt
exit

write mem

```

Post-deployment steps

When all subsystems are deployed, use the admin account to access the Cloud Manager administrative console and begin configuration. Optionally, take VMware snapshots of the VMs.

About this task

When you have deployed all required API Connect components, and all subsystems are running, complete the following:

Procedure

1. If you have not previously accessed the Cloud Manager, verify you can access the API Connect Cloud Manager. Enter the URL in your browser. The syntax is `https://<hostname.domain>/admin`. For example:

```
https://cloud-admin-ui.testsrv0231.subnet1.example.com/admin
```

The first time that you access the Cloud Manager user interface, you enter `admin` for the user name and `7iron-hide` for the password. You will be prompted to change the Cloud Administrator password and email address. See [Accessing the Cloud Manager user interface](#).

2. Define your API Connect configuration by using the API Connect Cloud Manager. See [Cloud Manager configuration checklist](#).
3. Optional: Take a Virtual Machine (VM) snapshot of all your VMs; see [Using VM snapshots for infrastructure backup and disaster recovery](#) for details. This action does require a brief outage while all of the VMs in the subsystem cluster are shut down - do not take snapshots of running VMs, as they might not restore successfully.

VM snapshots can offer a faster recovery when compared to redeploying OVAs and restoring from normal backups.

Important: VM snapshots are not an alternative to the standard API Connect backups. You should have completed an API Connect backup of each subsystem when verifying initial deployment, as advised in:

- [Verify installation of the Management subsystem](#)
- [Verifying deployment of the Developer Portal subsystem](#)
- [Verifying deployment of the Analytics subsystem](#)

Installing a two data center deployment on VMware

How to install a two data center disaster recovery (2DCDR) deployment on VMware.

Before you begin

Ensure that you understand the concepts of a 2DCDR deployment in API Connect. For more information, see [A two data center deployment strategy on VMware](#).

Review the information in [Requirements for initial deployment on VMware](#) and, if appropriate, [What's new for v10 installation from previous versions](#).

Familiarize yourself with the instructions in [First steps for deploying in a VMware environment](#). Complete these instructions with the additional steps in this topic that are specific to installing a 2DCDR deployment.

Restriction: It is not possible to use the Automated API behavior testing application ([Installing the Automated API behavior testing application](#)) in a 2DCDR configuration.

About this task

Installing API Connect as a 2DCDR deployment is similar to a single installation. The deployment in each data center is effectively an instance of the same API Connect deployment. The following endpoints must be the same on both data centers:

Management subsystem endpoints

- `platform-api`
- `consumer-api`
- `cloud-admin-ui`
- `api-manager-ui`

Portal subsystem endpoints

- `portal-admin`
- `portal-www`

To install a 2DCDR deployment, you follow the normal stand-alone API Connect installation steps, but before you generate the ISO files you must set some additional 2DCDR properties.

The 2DCDR properties are described in the following table.

Table 1. 2DCDR properties

Setting	Description
<code>multi-site-ha-enabled</code>	Indicates whether 2DCDR deployment is enabled. Set to <code>true</code> to enable 2DCDR. Set to <code>false</code> to disable 2DCDR. The default value is <code>false</code> .
<code>multi-site-ha-mode</code>	The 2DCDR mode of the data center. Possible values are: <ul style="list-style-type: none">• <code>active</code> - indicates the primary data center.• <code>passive</code> - indicates the warm-standby data center.
<code>management-replication</code>	Required only for the management subsystem. The external ingress name for the management subsystem in the current data center in the 2DCDR deployment. The name is a unique fully qualified hostname that the other data center uses to communicate with the current data center.
<code>portal-replication</code>	Required only for the portal subsystem. The external ingress name for the portal subsystem in the current data center in the 2DCDR deployment. The name is a unique fully qualified hostname that the other data center uses to communicate with the current data center.
<code>replication-peer-fqdn</code>	The ingress hostname for the other data center in the 2DCDR deployment. This information is required so that the two data centers can communicate with each other.
<code>siteName</code>	Unique descriptive name for the API Connect data center. Must be different in each data center. This name is used in the hostnames, and so can contain only <code>a-z</code> and <code>0-9</code> characters. You can set a <code>siteName</code> only when the subsystems are first deployed, and you cannot change the name after deployment. If you don't set <code>siteName</code> at first deployment, the <code>siteName</code> is autogenerated.

Important:

- Use a single APICUP project for all subsystems so that the API Connect deployments in both data centers have the same certificate chains. If network access to both data centers from the same APICUP project location is not possible, you can copy the project directory across to the other data center. If you are copying the project directory to your other data center, ensure that you keep backups of both project directories.
- The original project directory that is created with APICUP during the initial product installation is required to both restore the database and to upgrade your deployment. You cannot restore the database or complete an upgrade without the initial project directory because it contains pertinent information about the cluster.
- Subsystem endpoints for the components cannot be changed.
- The subsystem name of the management and portal subsystems must consist of lowercase alphanumeric characters or '-', contain no spaces, start with an alphabetic character, and end with an alphanumeric character.

Procedure

- **Installing on the active data center**

The following example shows how to configure the 2DCDR settings for deploying on the active data center (DC). In this example, the active data center is called **dallas** (DC1), and the warm-standby data center is called **raleigh** (DC2).

1. Complete the steps in [Configuring the Management subsystem](#) up to and including step [14](#) to configure the standard (non-2DCDR) settings for the management subsystem in Dallas (DC1).
2. Configure the 2DCDR values to set Dallas to be active for the management subsystem in DC1.

Run the following commands:

```
apicup subsys set mgmt_dallas multi-site-ha-enabled=true
apicup subsys set mgmt_dallas multi-site-ha-mode=active
apicup certs set mgmt_dallas management-replication-ingress --clear
apicup subsys set mgmt_dallas management-replication=mgrreplicationdallas.cluster1.example.com
apicup subsys set mgmt_dallas replication-peer-fqdn=mgrreplicationraleigh.cluster2.example.com
apicup subsys set mgmt_dallas site-name=dallas
```

where

- **mgmt_dallas** is the name of the management subsystem that you are configuring.
 - **mgrreplicationdallas.cluster1.example.com** is the ingress hostname for the current data center in the two data center deployment.
 - **mgrreplicationraleigh.cluster2.example.com** is the ingress hostname for the other data center in the two data center deployment.
 - **dallas** is the name of the site in the current data center.
3. Complete the remainder of the steps in [Configuring the Management subsystem](#) from step [15](#) to verify that the configuration settings are valid, create your ISO file, and deploy the OVF template.
 4. Complete the steps in [Configuring the Developer Portal subsystem](#) up to and including step [16](#) to configure the standard (non-2DCDR) settings for the portal subsystem in Dallas (DC1).
 5. Configure the 2DCDR values to set Dallas to be active for the portal subsystem in DC1.

Run the following commands:

```
apicup subsys set port_dallas multi-site-ha-enabled=true
apicup subsys set port_dallas multi-site-ha-mode=active
apicup certs set port_dallas portal-replication-ingress --clear
apicup subsys set port_dallas portal-replication=ptlreplicationdallas.cluster1.example.com
apicup subsys set port_dallas replication-peer-fqdn=ptlreplicationraleigh.cluster2.example.com
apicup subsys set port_dallas site-name=dallas
```

where

- **port_dallas** is the name of the portal subsystem that you are configuring.
 - **ptlreplicationdallas.cluster1.example.com** is the ingress hostname for the current data center in the two data center deployment.
 - **ptlreplicationraleigh.cluster2.example.com** is the ingress hostname for the other data center in the two data center deployment.
 - **dallas** is the name of the site in the current data center.
6. Complete the remainder of the steps in [Configuring the Developer Portal subsystem](#) from step [19](#) to verify that the configuration settings are valid, create your ISO file, and deploy the OVF template.
 7. Log in to the virtual machine subsystems on DC1 by using an SSH tool, and check the status of the installation by running the `apic status` command.
 8. Set your dynamic router to direct all traffic to DC1. The endpoints that must be directed to DC1 are:

Management subsystem endpoints

- **platform-api**
- **consumer-api**
- **cloud-admin-ui**
- **api-manager-ui**

Portal subsystem endpoints

- **portal-admin**
- **portal-www**

• Installing on the warm-standby data center

The following example shows how to set the 2DCDR values for the remote data center called **raleigh** (DC2), which is being installed as the warm-standby location.

Important: Remember to use the same APICUP project directory as for the active data center. If you cannot use the same project directory because its location does not have network access to both data centers, you can copy the project directory over. Ensure that backups are kept of both project directories in each data center.

1. Complete the steps in [Configuring the Management subsystem](#) up to and including step [14](#) to configure the standard (non-2DCDR) settings for the management subsystem in Raleigh (DC2).
2. Configure the 2DCDR values to set the management subsystem to be the warm-standby in the Raleigh (DC2) data center.

Run the following commands:

```
apicup subsys set mgmt_raleigh multi-site-ha-enabled=true
apicup subsys set mgmt_raleigh multi-site-ha-mode=passive
apicup certs set mgmt_raleigh management-replication-ingress --clear
apicup subsys set mgmt_raleigh management-replication=mgrreplicationraleigh.cluster2.example.com
apicup subsys set mgmt_raleigh replication-peer-fqdn=mgrreplicationdallas.cluster1.example.com
apicup subsys set mgmt_raleigh site-name=raleigh
```

where

- **mgmt_raleigh** is the name of the management subsystem that you are configuring.

- `mgrreplicationraleigh.cluster2.example.com` is the ingress hostname for the current data center in the two data center deployment.
 - `mgrreplicationdallas.cluster1.example.com` is the ingress hostname for the other data center in the two data center deployment.
 - `raleigh` is the name of the site in the current data center.
3. Set the encryption key manually for Raleigh (DC2) by copying it from Dallas (DC1).
Run the following commands:

```
apicup certs get mgmt_dallas encryption-secret -t key > mgmt-encryption-secret
apicup certs set mgmt_raleigh encryption-secret mgmt-encryption-secret
```

4. Check that the management encryption key is set the same on both DC1 and DC2.
Run the following commands, and verify that the outputs match:

```
apicup certs get mgmt_dallas encryption-secret -t key
apicup certs get mgmt_raleigh encryption-secret -t key
```

5. Complete the remainder of the steps in [Configuring the Management subsystem](#) from step 15 to verify that the configuration settings are valid, create your ISO file, and deploy the OVF template.
6. Complete the steps in [Configuring the Developer Portal subsystem](#) up to and including step 16 to configure the standard (non-2DCDR) settings for the portal subsystem in Raleigh (DC2).
7. Configure the 2DCDR values to set Raleigh to be warm-standby for the portal subsystem on DC2.
Run the following commands:

```
apicup subsys set port_raleigh multi-site-ha-enabled=true
apicup subsys set port_raleigh multi-site-ha-mode=passive
apicup certs set port_raleigh portal-replication-ingress --clear
apicup subsys set port_raleigh portal-replication=ptlreplicationraleigh.cluster2.example.com
apicup subsys set port_raleigh replication-peer-fqdn=ptlreplicationdallas.cluster1.example.com
apicup subsys set port_raleigh site-name=raleigh
```

where

- `port_raleigh` is the name of the portal subsystem that you are configuring.
 - `ptlreplicationraleigh.cluster2.example.com` is the ingress hostname for the current data center in the two data center deployment.
 - `ptlreplicationdallas.cluster1.example.com` is the ingress hostname for the other data center in the two data center deployment.
 - `raleigh` is the name of the site in the current data center.
8. Copy the encryption key for the portal subsystem on DC1 over to the portal subsystem on DC2.
Run the following command to get the encryption key for the portal subsystem on DC1, and put it into a file that is called `port-encryption-secret`:

```
apicup certs get port_dallas encryption-secret -t key > port-encryption-secret
```

Run the following command to use the `port-encryption-secret` file to set the same encryption key on DC2:

```
apicup certs set port_raleigh encryption-secret port-encryption-secret
```

Where `port_dallas` is the name of the portal subsystem on DC1, and `port_raleigh` is the name of the portal subsystem on DC2.

To check that the encryption key is set the same on both DC1 and DC2, you can run the following commands and verify that the outputs match:

```
apicup certs get port_dallas encryption-secret -t key
apicup certs get port_raleigh encryption-secret -t key
```

9. Complete the remainder of the steps in [Configuring the Developer Portal subsystem](#) from step 19 to verify that the configuration settings are valid, create your ISO file, and deploy the OVF template.
10. Log in to the virtual machine subsystems on DC2 by using an SSH tool, and check the status of the installation by running the `apic status` command.

• **Converting a single data center deployment to two data centers**

Important:

- Use the existing deployment as the active site. The new site must be set to warm-standby. If you set your existing deployment to warm-standby then all your management and portal data is erased.
- The ingress endpoint values from the existing deployment become the global ingress endpoints for the two center deployment. The following endpoints must be set the same on your new deployment, as they are on your existing deployment:

Management subsystem endpoints

- `platform-api`
- `consumer-api`
- `cloud-admin-ui`
- `api-manager-ui`

Portal subsystem endpoints

- `portal-admin`
- `portal-www`

- Configure your network to be able to route the endpoints to either data center, depending on which is the active.
- You cannot configure a `site-name` for the existing management or portal cluster because the `site-name` property can be configured only at first deployment.

The following example shows how to update the 2DCDR values to make the existing data center `Dallas` the active, and for installing and setting the new remote data center `Raleigh` as warm-standby.

1. Configure the 2DCDR values to set Dallas to be active for the API Manager subsystem on DC1.
Run the following commands:

```
apicup subsys set mgmt_dallas multi-site-ha-enabled=true
apicup subsys set mgmt_dallas multi-site-ha-mode=active
```

```

apicup certs set mgmt_dallas management-replication-ingress --clear
apicup subsys set mgmt_dallas management-replication=mgrreplicationdallas.cluster1.example.com
apicup subsys set mgmt_dallas replication-peer-fqdn=mgrreplicationraleigh.cluster2.example.com

```

where

- `mgmt_dallas` is the name of the management subsystem that you are configuring.
- `mgrreplicationdallas.cluster1.example.com` is the ingress hostname for the current data center in the two data center deployment.
- `mgrreplicationraleigh.cluster2.example.com` is the ingress hostname for the other data center in the two data center deployment.

Note: As the API Manager subsystem already exists, you cannot set the `site-name` property.

2. Configure the 2DCDR values to set Dallas to be active for the portal subsystem on DC1.

Run the following commands:

```

apicup subsys set port_dallas multi-site-ha-enabled=true
apicup subsys set port_dallas multi-site-ha-mode=active
apicup certs set port_dallas portal-replication-ingress --clear
apicup subsys set port_dallas portal-replication=ptlreplicationdallas.cluster1.example.com
apicup subsys set port_dallas replication-peer-fqdn=ptlreplicationraleigh.cluster2.example.com

```

where

- `port_dallas` is the name of the portal subsystem that you are configuring.
- `ptlreplicationdallas.cluster1.example.com` is the ingress hostname for the current data center in the two data center deployment.
- `ptlreplicationraleigh.cluster2.example.com` is the ingress hostname for the other data center in the two data center deployment.

Note: As the portal subsystem already exists, you cannot set the `site-name` property.

3. Run `apicup` to update the settings in each subsystem, for example:

```
apicup subsys install subsystem_name
```

where `subsystem_name` is the name of the management or portal subsystem that you want to update, for example `mgmt_dallas`.

4. Log in to the virtual machine subsystems on DC1 by using an SSH tool, and check the status of the installation by running the `apic status` command.
5. Add the new DC, Raleigh (DC2) by completing the steps in [Configuring the Management subsystem](#) up to and including step 14 to create a management subsystem in Raleigh (DC2).

Note:

- Remember to use the same APICUP project directory as for the active data center.
- Remember to set the same endpoints as for DC1:

Management subsystem endpoints

- `platform-api`
- `consumer-api`
- `cloud-admin-ui`
- `api-manager-ui`

Portal subsystem endpoints

- `portal-admin`
- `portal-www`

6. Configure the 2DCDR values to set Raleigh to be warm-standby for the API Manager subsystem on DC2.

Run the following commands:

```

apicup subsys set mgmt_raleigh multi-site-ha-enabled=true
apicup subsys set mgmt_raleigh multi-site-ha-mode=passive
apicup certs set mgmt_raleigh management-replication-ingress --clear
apicup subsys set mgmt_raleigh management-replication=mgrreplicationraleigh.cluster2.example.com
apicup subsys set mgmt_raleigh replication-peer-fqdn=mgrreplicationdallas.cluster1.example.com
apicup subsys set mgmt_raleigh site-name=raleigh

```

where

- `mgmt_raleigh` is the name of the management subsystem that you are configuring.
- `mgrreplicationraleigh.cluster2.example.com` is the ingress hostname for the current data center in the two data center deployment.
- `mgrreplicationdallas.cluster1.example.com` is the ingress hostname for the other data center in the two data center deployment.
- `raleigh` is the name of the site in the current data center.

Note: As this is a new deployment of the API Manager subsystem, you can set the `site-name` property.

7. Set the encryption key manually for Raleigh (DC2) by copying it from Dallas (DC1).

Run the following commands:

```

apicup certs get mgmt_dallas encryption-secret -t key > mgmt-encryption-secret
apicup certs set mgmt_raleigh encryption-secret mgmt-encryption-secret

```

8. Check that the management encryption key is set the same on both DC1 and DC2.

Run the following commands, and verify that the outputs match:

```

apicup certs get mgmt_dallas encryption-secret -t key
apicup certs get mgmt_raleigh encryption-secret -t key

```

9. Complete the remainder of the steps in [Configuring the Management subsystem](#) from step 15 to verify that the configuration settings are valid, create your ISO file, and deploy the OVF template.
10. Complete the steps in [Configuring the Developer Portal subsystem](#) up to and including step 16 to create a portal subsystem in Raleigh (DC2).
11. Configure the 2DCDR values to set Raleigh to be warm-standby for the portal subsystem on DC2.

Run the following commands:

```
apicup subsys set port_raleigh multi-site-ha-enabled=true
apicup subsys set port_raleigh multi-site-ha-mode=passive
apicup certs set port_raleigh portal-replication-ingress --clear
apicup subsys set port_raleigh portal-replication=ptlreplicationraleigh.cluster2.example.com
apicup subsys set port_raleigh replication-peer-fqdn=ptlreplicationdallas.cluster1.example.com
apicup subsys set port_raleigh site-name=raleigh
```

where

- `port_raleigh` is the name of the portal subsystem that you are configuring.
- `ptlreplicationraleigh.cluster2.example.com` is the ingress hostname for the current data center in the two data center deployment.
- `ptlreplicationdallas.cluster1.example.com` is the ingress hostname for the other data center in the two data center deployment.
- `raleigh` is the name of the site in the current data center.

12. Run the following command to generate the default portal certificates on the warm-standby site:

```
apicup certs generate port_raleigh
```

13. Copy the encryption key for the portal subsystem on DC1 over to the portal subsystem on DC2.

Run the following command to get the encryption key for the portal subsystem on DC1, and put it into a file that is called `port-encryption-secret`:

```
apicup certs get port_dallas encryption-secret -t key > port-encryption-secret
```

Run the following command to use the `port-encryption-secret` file to set the same encryption key on DC2:

```
apicup certs set port_raleigh encryption-secret port-encryption-secret
```

Where `port_dallas` is the name of the portal subsystem on DC1, and `port_raleigh` is the name of the portal subsystem on DC2.

To check that the encryption key is set the same on both DC1 and DC2, you can run the following commands and verify that the outputs match:

```
apicup certs get port_dallas encryption-secret -t key
```

```
apicup certs get port_raleigh encryption-secret -t key
```

14. Complete the remainder of the steps in [Configuring the Developer Portal subsystem](#) from step 19 to verify that the configuration settings are valid, create your ISO file, and deploy the OVF template.

15. Log in to the virtual machine subsystems on DC2 by using an SSH tool, and check the status of the installation by running the `apic status` command.

16. Set your dynamic router to direct all traffic to DC1. The endpoints that must be directed to DC1 are:

Management subsystem endpoints

- `platform-api`
- `consumer-api`
- `cloud-admin-ui`
- `api-manager-ui`

Portal subsystem endpoints

- `portal-admin`
- `portal-www`

- **Validating your two data center deployment**

Use the `apicup health-check` command to confirm that the management subsystems are ready and synchronized:

```
apicup subsys health-check <management subsystem>
```

If the management subsystem is healthy, the `health-check` command returns silently with no output.

Create test data on both management and portal subsystems, and complete a test failover:

1. Create some test data, for example, on management create a local user registry: [Configuring a local user registry](#).
2. Complete a failover: [Failure handling of a two data center deployment](#).
3. Verify that the test data created on the original active site, exists on the new active site. For example, login to the Cloud Manager UI on the new active site and confirm the presence of the test local user registry.
4. Delete the test data.
5. Complete another failover to return to your original active and warm-standby configuration.

Example

Example of the management subsystem values that you can view by running the `apicup subsys get mgmt_name` command:

```
apicup subsys get mgmt_dallas
```

```
Appliance settings
```

```
=====
```

Name	Value
Description	
-----	-----

additional-cloud-init-file	
(Optional) Path to additional cloud-init yml file	
data-device	sdb
VM disk device (usually `sdb` for SCSI or `vdb` for VirtIO)	
default-password	
\$6\$rounds=4096\$vtcqpAVK\$dzqrOeYP33WTvTug38Q4R1d518TmdQgezTnkX/PFwkzTiZ2S0CqNrr1S4b08tOc4p.OEg4BtzBe/r8RAk.gw/	(Optional)
Console login password for `apicadm` user, password must be pre-hashed	
dns-servers	[8.8.8.8]
List of DNS servers	

```

extra-values-file
(Optional) Path to additional configuration yml file
k8s-pod-network      172.16.0.0/16
(Optional) CIDR for pods within the appliance
k8s-service-network 172.17.0.0/16
(Optional) CIDR for services within the appliance
public-iface         eth0
Device for API/UI traffic (Eg: eth0)
search-domain        [subnet1.example.com]
List for DNS search domains
ssh-keyfiles         [/home/vsphere/.ssh/id_rsa.pub]
List of SSH public keys files
traffic-iface        eth0
Device for cluster traffic (Eg: eth0)

```

Subsystem settings
=====

Name	Value
Deployment profile	nlxc4.m16
Deployment profile (nlxc2.m16/n3xc4.m16) for analytics, (nlxc4.m16/n3xc4.m16) for management, (nlxc2.m8/n3xc4.m8) for portal	
license-use	production
License use (production/nonproduction)	
multi-site-ha-enabled	true
Multi site HA enabled	
multi-site-ha-mode	active
Multi site HA mode (active/passive)	
replication-peer-fqdn	mgrreplicationraleigh.cluster2.example.com
Replication peer fully qualified name (replication endpoint of active mode site)	
site-name	dallas
Site name, used in k8s resource names	
test-and-monitor-enabled	false
Test and Monitor enabled	

Endpoints
=====

Name	Value
api-manager-ui	api-manager-ui.testsrv0231.subnet1.example.com
FQDN of API manager UI endpoint	
cloud-admin-ui	cloud-admin-ui.testsrv0231.subnet1.example.com
FQDN of Cloud admin endpoint	
consumer-api	consumer-api.testsrv0231.subnet1.example.com
FQDN of consumer API endpoint	
hub	
FQDN of Test and Monitor hub endpoint, only required if Test and Monitor is enabled	
management-replication	mgrreplicationdallas.cluster1.example.com
FQDN of Management replication endpoint, only required if HA is enabled	
platform-api	platform-api.testsrv0231.subnet1.example.com
FQDN of platform API endpoint	
turnstile	
FQDN of Test and Monitor turnstile endpoint, only required if Test and Monitor is enabled	

Example of the portal subsystem values that you can view by running the `apicup subsys get port_name` command:

```
apicup subsys get port_dallas
```

Appliance settings
=====

Name	Value
additional-cloud-init-file	
(Optional) Path to additional cloud-init yml file	
data-device	sdb
VM disk device (usually `sdb` for SCSI or `vdb` for VirtIO)	
default-password	
\$6\$rounds=4096\$vtcqpAVK\$dzqrOeYP33WTvTug38Q4Rld518TmdQgezzTnkX/PFwkzTiZ2S0CqNRr1S4b08tOc4p.OEg4BtzBe/r8RAk.gW/	(Optional)
Console login password for `apicadm` user, password must be pre-hashed	
dns-servers	[192.168.1.201]
List of DNS servers	
extra-values-file	
(Optional) Path to additional configuration yml file	
k8s-pod-network	172.16.0.0/16
(Optional) CIDR for pods within the appliance	
k8s-service-network	172.17.0.0/16
(Optional) CIDR for services within the appliance	
public-iface	eth0
Device for API/UI traffic (Eg: eth0)	
search-domain	[subnet1.example.com]
List for DNS search domains	
ssh-keyfiles	[/home/vsphere/.ssh/id_rsa.pub]
List of SSH public keys files	

```

traffic-iface          eth0
Device for cluster traffic (Eg: eth0)

Subsystem settings
=====
Name                  Value
Description           -----
-----
deployment-profile    n1xc2.m8
Deployment profile (n1xc2.m16/n3xc4.m16) for analytics, (n1xc4.m16/n3xc4.m16) for management, (n1xc2.m8/n3xc4.m8) for portal
license-use           production
License use (production/nonproduction)
multi-site-ha-enabled true
Multi site HA enabled
multi-site-ha-mode    active
Multi site HA mode (active/passive)
replication-peer-fqdn ptlreplicationraleigh.cluster2.example.com
Replication peer fully qualified name (replication endpoint of active mode site)
site-name             dallas
Site name, used in k8s resource names

Endpoints
=====
Name                  Value
Description           -----
-----
portal-admin          portal-api.example.com
FQDN of Portal admin endpoint
portal-replication    ptlreplicationdallas.cluster1.example.com
FQDN of Portal replication endpoint, only required if HA is enabled
portal-www            portal-www.example.com
FQDN of Portal web endpoint

```

Related concepts

- [Maintaining a two data center deployment](#)

Related tasks

- [Failure handling of a two data center deployment](#)
- [Recovering from a failover of a two data center deployment](#)

Upgrading API Connect

Upgrade API Connect on VMware to a newer version.

After reviewing the overview, use the procedures in this section to upgrade API Connect on VMware.

Note:

- For upgrading from v5, see [Migrating a Version 5 deployment](#).
- For information about how to maintain a two data center deployment, including upgrade considerations, see [Maintaining a two data center deployment](#).
- [Upgrade considerations on VMware](#)
Before upgrading API Connect on VMware, ensure that your deployment meets all the upgrade requirements.
- [Preparing to upgrade on VMware](#)
Prepare your VMware deployment for upgrading to the new version of API Connect 10.0.5.x.
- [Upgrading to the latest release on VMware](#)
You can upgrade API Connect on VMware.
- [Upgrading DataPower Gateway Service](#)
Use these instructions when upgrading DataPower Gateway Service in a non-Kubernetes environment, during upgrade of API Connect.
- [Upgrading from 2018 on VMware](#)
You can upgrade API Connect on VMware from the latest release of v2018 to the latest release of v10.
- [Enabling Analytics after upgrading](#)
After the upgrade successfully completes, re-enable Analytics data collection.
- [Troubleshooting upgrades on VMware](#)
Review the following troubleshooting guidance if you encounter a problem while upgrading API Connect on VMware.

Upgrade considerations on VMware

Before upgrading API Connect on VMware, ensure that your deployment meets all the upgrade requirements.

Contents:

- [Supported upgrade paths](#)

- [Upgrade requirements](#)
- [Requirement: Validating an API definition file with the API Dev Tools Swagger Parser](#)
- [Limitation: x-ibm-gateway-cors-allow-credentials-when-no-cors-policy API property is deprecated](#)

Tip: After upgrade, clear your browser cache, and open a new browser window. This action avoids stale cache issues in your browser, that can result in unexpected behavior in the Cloud Manager and API Manager UIs.

Supported upgrade paths

Restriction: At this time, you cannot upgrade directly to 10.0.5.3 or later from 10.0.1.7-eus or 10.0.1.8-eus. See Table 1 for alternative upgrade paths.

Table 1. Supported upgrade paths

Upgrade from:	How to upgrade to 10.0.5.4:
<ul style="list-style-type: none"> • 10.0.5.3 • 10.0.5.2 • 10.0.5.1 • 10.0.4.0 and iFixes • 10.0.3.0 and iFixes • 10.0.1.12-eus • 10.0.1.11-eus • 10.0.1.9-eus • 10.0.1.6-eus • 10.0.1.5-eus and iFixes • 10.0.1.4-eus and iFixes • 10.0.1.2-eus and iFixes 	See Upgrading to the latest release on VMware Note: Version 10.0.5.0 did not support VMware.
<ul style="list-style-type: none"> • 10.0.1.8-eus • 10.0.1.7-eus 	Complete 2 upgrades: <ol style="list-style-type: none"> 1. First, upgrade to 10.0.1.9-eus as explained in Upgrading to 10.0.1.9-eus on VMware in the API Connect 10.0.1.x-eus documentation. 2. Then, upgrade to 10.0.5.4 as explained in Upgrading to the latest release on VMware in this documentation.
Older versions of 10.0.x: <ul style="list-style-type: none"> • 10.0.2.0 • 10.0.1.1 • 10.0.1.0 • 10.0.0.0 	Complete 2 upgrades: <ol style="list-style-type: none"> 1. First, upgrade to 10.0.1.2-ifix1-eus as explained in Upgrading from Version 10 to 10.0.1.2-ifix1-eus in the API Connect 10.0.1.x-eus documentation. To upgrade to 10.0.1.2-ifix1-eus, you must deploy a fresh cluster and restore existing data into it. This procedure is required to deploy VMs with Ubuntu 20.4. 2. Then, upgrade to 10.0.5.4 as explained in Upgrading to the latest release on VMware in this documentation.
2018 FP24 or later	See Upgrading from 2018 on VMware

Upgrade requirements

- API Connect is not supported on a FIPS-enabled environment.
- The instructions for upgrading apply to the latest version of API Connect.
- The upgrade order for subsystems is important. Upgrade the subsystems in the following order:
 1. Management
 2. Portal
 3. Analytics
 4. Gateway

Gateway must be upgraded after Management, because upgrading the Management service before the Gateway ensures that any new policies and capabilities will be available to a previously registered Gateway service.
- When maintaining API Connect, do not use `kubect1 exec` commands to access API Connect pods unless advised by IBM. Do not make any changes on the deployed VMs unless documented here or otherwise advised by IBM. Attempting to manually update packages, adding new users, or installing new software will likely cause problems. Operating system updates are handled by API Connect fix packs.

Validating an API definition file with the API Dev Tools Swagger Parser

From IBM® API Connect Version 10.0.3.0 or later, SwaggerDoc (OpenAPI 2.0) and OpenAPI 3.0 documents that form API definition files are now validated at stage or publish time by the API Dev Tools Swagger Parser. Before upgrading, you should ensure that your existing API definition files can be parsed and are considered valid by the Swagger Parser tool. Note that any API definitions that are already staged or published are not impacted, but would undergo Swagger Parser validation again should they be re-staged or re-published in the future.

You can use either of the following techniques to ensure that existing API definition files are can be parsed and considered valid by the Swagger Parser:

- Download and install the `swagger-parser-cli` tool (see <https://www.npmjs.com/package/swagger-cli>). This tool includes a `validate` command that can validate local API definition files against the SwaggerDoc (OpenAPI 2.0) and OpenAPI 3.0 schemas, in both JSON and YAML formats.
- Use the browser-based Swagger Parser Validator available at <https://apitools.dev/swagger-parser/online/>; this accepts a URL to existing files available online, or can accept text copied and pasted into the tool, in both JSON and YAML formats. It can likewise validate against the SwaggerDoc (OpenAPI 2.0) and OpenAPI 3.0 schemas.

Note: Staging or publishing an API in API Connect results in additional syntactic validation of IBM extensions to SwaggerDoc (OpenAPI 2.0) and OpenAPI 3.0, as well as semantic validation to make sure that policies and other resources that the API refers to are valid within the context of the staging or publishing operation. Any errors

associated with this additional validation will not be caught by the local or online Swagger Parser tool, nor will they be caught by the `apic validate` toolkit CLI command.

The Swagger Parser tool will report an invalid OpenAPI document. For example, if a document is missing a mandatory section such as `info` or `paths`, the following error is reported:

```
[object Object] is not a valid Swagger API definition
```

If a required property is missing from a given section, a specific error is reported. For example, if the `info` section is missing the required `version` property, the following error is reported:

```
Swagger schema validation failed.  
Missing required property: version at #/info
```

```
JSON_OBJECT_VALIDATION_FAILED
```

Additionally, if the document contains any local references (that is, values of `$ref` properties) that do not resolve to an existing reference within the document (known as de-referencing), this is also be flagged as an error. For example, if a document contains the following as a reference:

```
schema:  
  $ref: '#/definitions/Pet'
```

then the definition of `Pet` (that is, `#/definitions/Pet`) **must** exist in the same document, at the given path (`definitions` section in this example), similar to the following:

```
definitions:  
  Pet:  
    type: object  
    required:  
    - name  
    properties:  
      id:  
        type: integer  
        format: int64
```

If the local reference is not found within the document, the following de-referencing error is reported:

```
Token "Pet" does not exist.
```

x-ibm-gateway-cors-allow-credentials-when-no-cors-policy API property is deprecated

In v2018, when cross-origin resource sharing (CORS) is enabled for an API, every response to a CORS request contains the header:

```
Access-Control-Allow-Credentials: true
```

Inclusion of Access-Control-Allow-Credentials (ACAC) does not provide optimal security, so starting with v2018.4.1.17, an API property can be set to prevent this header from being returned:

```
Name: x-ibm-gateway-cors-allow-credentials-when-no-cors-policy  
Collection: *  
Value (string): off
```

The `Collection` property refers to a specific Catalog to which a property value applies. The "*" value means the value applies to all catalogs. You can either specify * or a Catalog (collection) name.

The property can also be set by using the DataPower CLI:

```
api-definition myAPI  
  property x-ibm-gateway-cors-allow-credentials-when-no-cors-policy * off  
exit
```

Important: The value of this property must be exactly the string `off` (case-sensitive) in order to remove the header.

It is only possible to turn off this header, or leave it on, for an entire API. In order to have more control, such as returning the header for some origins and not others, you must upgrade to v10 and use the CORS policy feature. For more info, see:

- [Enabling CORS support for an API](#)
- OpenAPI 3.0 editor: [Setting API properties](#)
- OpenAPI 2.0 editor: [Setting API properties](#)

In API Connect 10.0.3.0 or later, the `x-ibm-gateway-cors-allow-credentials-when-no-cors-policy` property works, but only for API definitions which do NOT contain a cors-policy. You cannot define both this property and a CORS Policy on the same API definition.

This property is deprecated in v10: a warning will be written to the log for every CORS request received by an API whose definition contains this property.

For v10, the recommended way to ensure the ACAC header does not appear in CORS responses is to add a CORS policy to the API definition. In v10, any API definition which has a cors-policy defined will NOT return the ACAC header by default. Note that it is possible in the CORS policy rules to explicitly return the ACAC header for some or all origins.

If you move your configuration from Version 2018 to 10.0.3.0 or later, and have this custom property defined in your API definitions, you will not see any change in behavior, other than new warnings in the log. This property will remove the ACAC header for any API definition where it appears.

If, when you move your configuration from Version 2018 to Version 10, you want to have the ACAC header present in the CORS responses, you can leave the API definitions as-is (without the new property defined) and the ACAC header will be returned, as long as a cors-policy is not added to the API definition. Note also that even with a CORS policy defined for an API, the ACAC header can be returned if desired by explicitly allowing it in the rules of the CORS policy.

Note that the `x-ibm-gateway-cors-allow-credentials-when-no-cors-policy` property in an API definition:

- Will cause a commit-time error in v10 if a cors-policy is also present in the same API definition

- Will NOT work in v10 or v2018 to turn ON the ACAC header; it can ONLY be used to turn OFF the header. Explanation:
 - In v2018, ACAC is on by default
 - In v10, without cors-policy, ACAC is on by default
 - In v10, with cors-policy, ACAC can be turned on for specific origins via rules in cors-policy
- [Additional considerations for upgrading analytics from a version prior to v10.0.5.0](#)
API Connect v10.0.5.0 introduced a new architecture for the Analytics subsystem, so some additional steps are required before and after upgrading from API Connect 10.0.4-ifix3 or 10.0.1.x.

Additional considerations for upgrading analytics from a version prior to v10.0.5.0

API Connect v10.0.5.0 introduced a new architecture for the Analytics subsystem, so some additional steps are required before and after upgrading from API Connect 10.0.4-ifix3 or 10.0.1.x.

The key changes to the Analytics subsystem are highlighted in [What's new in Version 10.0.5](#).

The most important consideration is that existing analytics data is deleted during the upgrade from 10.0.4-ifix3, and from 10.0.1.7-eus (or higher), and cannot be restored. After upgrading to v10.0.5.0 or later, you will have an empty analytics database. If you want to keep your existing analytics data on an external system, there are three options:

- [Exporting analytics data from the UI](#)
Before upgrading to v10.0.5, export any analytics data that you wish to keep using the UI.
- [Exporting analytics with the REST API](#)
Before upgrading to v10.0.5, export any analytics data that you want to keep by using the REST API.
- [Offloading analytics data](#)
An alternative to exporting your analytics data before upgrading to v10.0.5 is to enable analytics offload. However, when enabled this feature only offloads new analytics data. But if your upgrade is not planned for a few days or weeks, you can enable this feature now and avoid the need to export any new analytics data.
- [Update the analytics extra-values file](#)
Review all extra-values file settings and ensure that the file contains only valid v10.0.5 settings.

Exporting analytics data from the UI


Before upgrading to v10.0.5, export any analytics data that you wish to keep using the UI.

About this task

Note: This page refers to analytics features that existed in pre-v10.0.5 releases, for the purposes of exporting analytics data before upgrade. Exporting analytics data from v10.0.5 is described here [Exporting analytics from v10.0.5](#)
Use the Analytics Discover tab to search for event data and then export the *hits* (the records in the search results) to a file. Each hit represents one API invocation in the Gateway server. When you view the hits in the UI, the display is limited to 500 records. To view the entire set of hits, export the data to a CSV or JSON Lines file.

Procedure

1. Navigate to the Analytics page and click Discover.
2. Run a search.

Select a predefined search to run, or click  to define your own search, and then run it. Optionally apply filters to refine the results. The export option is only available when a search returns at least one hit.

3. Click Export (this option displays before the search results).
4. In the "Export Hits" panel, select the Filetype for the exported data.

There are two options for the file's type:

- a. JSON Lines (.txt): Exports data to a .txt file that contains a complete JSON record for each hit.
- b. Comma-separated Values (.csv): Exports to a .csv file that can be viewed either in a text editor, a spreadsheet, or some other CSV viewer.
If you intend to view the .csv file in a viewer that does not support special characters, select the following options as needed to ensure compatibility with your viewer:
 - i. Escape double quotes: Insert an additional quotation mark (") as an escape character before each " that is already included in the file.
 - ii. Escape special characters: Insert a quotation mark (") as an escape character before each of the following characters: = _ + % @

The content of the API event records in the CSV file depends on the logging policy that is set for the operation. If the logging policy includes the HTTP headers and the payload, then these details are included in the API event record. Where the payload is large, it might take longer to complete the export of the analytics data. For more information, see [activity-log.policy](#). For information about the fields that are included in the CSV file, see [API event records](#).

5. In the panel, click Export to start the export operation.
6. When the "The export has completed successfully." message displays, click Download to save the file, or click Another one? to run a new export operation.
(Remember to save the current export results before running a new export).

Attention:

- If your browser is configured to block pop-ups, the download might be blocked. The precise behavior varies across browsers, but if you see a notification about "blocked pop-ups", reconfigure the browser to always allow pop-ups for the API Manager host address. Then check the "Export Hits" panel for the Download option, which completes the operation. If you do not see a Download option, repeat the export operation.
- If you navigate away from the Analytics interface before the exported file is downloaded, the download operation is suspended. If you return to Analytics before any other export operation begins and before the file expires, you might see a Download option that allows you to finish downloading the file. Otherwise, you must repeat the entire export operation.
- Each downloaded file is saved to the download location that is configured for your browser, using a randomly generated file name. If you intend to export more than once, and you are exporting to the same location each time, you might want to rename each file with a meaningful name.

Exporting analytics with the REST API

Before upgrading to v10.0.5, export any analytics data that you want to keep by using the REST API.

Before you begin

API Connect REST API calls are authenticated by using a bearer token in the authorization header. Use the toolkit credentials to request a bearer token. Toolkit credentials can be found in the API Manager UI:

1. Log in to the API Manager UI.
2. From the home page, click the Download toolkit tile.
3. Download the Toolkit credentials.
4. Open the downloaded `credentials.json` file, example:

```
...
"toolkit": {
  "endpoint": "https://mgmt.api.example.com/api",
  "client_id": "7409693f-f726-48b4-8909-7c0d26f13e81",
  "client_secret": "5feeb0be-17f8-41a4-96d6-d40f33d69ef6"
},
...
```

Take note of the `toolkit.client_id`, `toolkit.client_secret`, and `toolkit.endpoint` from this file. You will use these to get your bearer token.

About this task

The Analytics REST API is documented in the [API Connect v10.0.5LATEST](#) API Explorer. You can export the data catalog-by-catalog using `/analytics/catalogs/{org}/{catalog}/analytics-services/{analytics-service}/events`

Procedure

1. Use your provider organization credentials and client_id/secret to request a bearer token:

```
curl -v -k -X POST -d '{"username": "<username>", "password": "<password>", "realm": "<realm>", "client_id": "<client_id>", "client_secret": "<client_secret>", "grant_type": "password"}' -H 'Content-Type: application/json' -H 'Accept: application/json' https://<management server platform api endpoint>/token
```

Where:

- `<username>` is the provider org owner.
- `<password>` is the password for the specified `<username>`.
- `<realm>` is the user registry realm for the specified `<username>`. The default provider organization user registry realm is `provider/default-idp-2`.
- `<client_id>` is the client ID from the "toolkit" section of the `credentials.json` file, or as provided by your API Connect cloud administrator.
- `<client_secret>` is the client secret from the "toolkit" section of the `credentials.json` file, or as provided by your API Connect cloud administrator.
- `<management server platform api endpoint>` is the platform API endpoint from the "toolkit" section of the `credentials.json` file, or as provided by your API Connect cloud administrator.

The bearer token is returned in the `access_token` property:

```
{
  "access_token": "<bearer token>",
  "token_type": "Bearer",
  "expires_in": 28800
}
```

Note the `expires_in` value, which is the number of seconds before the token expires. After expiry you need to request a new bearer token.

2. Use the returned bearer token to call the analytics REST API:

```
curl -v -k -H 'Accept: application/json' -H 'Authorization: Bearer <bearer token>' -X GET --url 'https://<management server platform api endpoint>/analytics/catalogs/<provider org name>/<catalog name>/analytics-services/<analytics service name>/events?after=now-7d'
```

Where:

- `<management server platform api endpoint>` is the `toolkit.endpoint`, but with the `/api` at the end replaced with `/analytics` to access the analytics APIs.

In this example, the REST API call returns the last 7 days (note the `?after=now-7d` at end of the URL) of event data for the specified catalog:

```
{"total":2,"found":true,"events":[{"id":"AYGp1P_4umraqLKQkId3","event":{"immediate_client_ip":"9.145.95.209","developer_org_name":"","resource_id":"","GET:/hola-mundo","developer_org_id":"","log_policy":"activity"...
```

Note: For large data sets it might be necessary to make the REST call repeatedly, specifying a different time range for each call. For example, to capture events logged during a specific 30-minute interval, use: `?after=2022-06-28T09:00:00&before=2022-06-28T09:30:00`

3. Additional filters can be used as described in the REST API documentation for the call to `GET /analytics/catalogs/{org}/{catalog}/analytics-services/{analytics-service}/events`

Offloading analytics data

An alternative to exporting your analytics data before upgrading to v10.0.5 is to enable analytics offload. However, when enabled this feature only offloads new analytics data. But if your upgrade is not planned for a few days or weeks, you can enable this feature now and avoid the need to export any new analytics data.

The steps for configuring analytics offload are covered in the v10.0.4 documentation: <https://www.ibm.com/docs/en/api-connect/10.0.x?topic=cyt-offloading-data-third-party-system>

Update the analytics extra-values file

Review all extra-values file settings and ensure that the file contains only valid v10.0.5 settings.

The features and settings that are configured with the extra-values file are different in v10.0.5 from previous releases. Review the contents of your analytics extra-values file, and ensure that it contains only settings applicable to v10.0.5: [Reference: Fields in analytics-extra-values](#).

Important: If your extra-values file contains a **template** section, this must be removed completely. If you have analytics memory overrides configured in a **template** section, then review the available [development profiles](#) for v10.0.5, and update your deployment profile to match your memory requirements: [Changing deployment profiles on VMware](#).

Preparing to upgrade on VMware

Prepare your VMware deployment for upgrading to the new version of API Connect 10.0.5.x.

Before you begin

Review the [Upgrade considerations on VMware](#) to ensure that you are following a supported upgrade path and that you understand important changes that might affect your deployment.

About this task

Complete the following steps to prepare your deployment for upgrading to the newest version of API Connect.

Procedure

1. Remove obsolete files to ensure a clean upgrade.

Attention: If you do not remove the obsolete files, the upgrade might fail.

- a. SSH into the node:

- i. Run the following command to connect as the API Connect administrator, replacing **ip_address** with the appropriate IP address:

```
ssh ip_address -l apicadm
```

- ii. When prompted, select Yes to continue connecting.

- iii. When you are connected, run the following command to receive the necessary permissions for working directly on the appliance:

```
sudo -i
```

- b. Check the `/etc/apt/sources.list.d` directory for files of the format: `repo-<subsystem>-<version>-<appliance version>.list`

- c. Delete all files with names that match the format in substep b.

Run the following command for each file to delete it:

```
rm <file_path>
```

2. If you are upgrading from a release earlier than 10.0.5.1, preserve your analytics data and review your analytics extra-values file.

If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.6-eus (or higher): Upgrading to 10.0.5.x results in the deletion of existing analytics data. If you want to retain your analytics data, then you must export it before upgrade. Some of the settings in the extra-values file are not compatible with 10.0.5, so they must be reviewed before upgrade is started. For more information, see [Additional considerations for upgrading analytics from a version prior to v10.0.5.0](#).

3. If you are upgrading from 10.0.5.1, check the version of **containerd** used in your deployment.

There is a known issue where upgrades from 10.0.1.7-eus or 10.0.1.8-eus to 10.0.5.1 incorrectly upgraded the **containerd** version. This did not cause a problem in later 10.0.1.x versions or in 10.0.5.1, but might cause problems in newer versions of API Connect. Correct the problem now to ensure clean upgrades in the future.

On each node, complete the following steps to determine the version of **containerd** and if necessary, downgrade it to the correct version:

- a. Run the following command to check the version of **containerd** used in your deployment:

```
kubectl get nodes -o wide
```

The response looks like the following example:

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION	CONTAINER-RUNTIME
s2toiapi05101	Ready	master	352d	v1.23.5	10.98.128.7	<none>	Ubuntu 20.04.5 LTS	5.4.0-125-generic	containerd://1.6.6

- If the response shows `containerd://1.5.11` for the CONTAINER-RUNTIME, skip to the next step because you are already using the correct version of **containerd**.
- If the response shows `containerd://1.6.6` for the CONTAINER-RUNTIME, complete this step to downgrade it to version 1.5.1.11.

- b. Run the following command on the node and verify that it returns successfully:

```
apic health-check
```

For more information on the health check, see [Checking cluster health on VMware](#).

- c. Run the following set of commands on the healthy node to downgrade the version of **containerd**:

```

systemctl stop appliance-manager
systemctl stop kubelet
systemctl stop containerd
apt-get --allow-downgrades upgrade -y containerd.io
sed -i 's/KillMode=process/KillMode=mixed/g' /lib/systemd/system/containerd.service
systemctl daemon-reload
systemctl restart containerd
apic lock
apic unlock

```

d. Run the following command to verify that `containerd` now displays `containerd://1.5.11` for the CONTAINER-RUNTIME:

```
kubectl get nodes -o wide
```

Remember to complete these steps on every node.

4. If you did not need to run a health check in the previous step, run it now.

```
apic health-check
```

A health check is run automatically during the upgrade, but you might want to run it now to ensure that you are backing up a healthy deployment in the next step. For more information on the health check, see [Checking cluster health on VMware](#).

5. Back up the API Connect deployment.

This manual backup ensures that any data changes that occurred after the most recent scheduled backup are captured.

For details on creating a backup, see the appropriate documentation for the version of API Connect that you are upgrading from:

- 10.0.5.x: [Backing up and restoring on VMware](#)
- 10.0.1.x: [Backing up and restoring](#)
- 10.0.4-ifix3: [Backing up and restoring on VMware](#)

6. If you are upgrading from 10.0.5.1 and there is a Helm instance of the fluent-bit service running, uninstall the service by running the following command:

```
helm uninstall fluent-bit
```

In new versions of API Connect, fluent-bit is installed as part of the upgrade process.

7. [Configuring remote logging for a VMware deployment](#).

If not already configured, send your deployment logs to a remote server. The upgrade process terminates pods, and the logs will be lost if not stored remotely.

8. Ensure that port 9178 is open between the Analytics, Management, and Portal subsystems, and that the deployment is running `apicup`.

An i/o timeout error can occur if the firewall rejects traffic to the appliance. Example error message:

```

apicup subsys install test-analytics upgrade_analytics_lts_v10.0.1.0.tgz
INFO[0000] Preparing update
INFO[0010] Sending update packet to cluster (connected to testsrv0192.subnet1.example.com)
Error: failed to install the subsystem: unable to open file send stream:
rpc error: code = Unavailable desc = all SubConns are in TransientFailure,
latest connection error: connection error:
desc = "transport: Error while dialing dial tcp 1.2.3.4:9178: i/o timeout"

```

If you receive the error, make sure port 9178 is open and try again.

9. Download the `apicup` command executable from the same URL where you downloaded files for the new version of API Connect.

You must use the latest `apicup` with the latest product version.

10. Ensure that the original project directory (created with the `apicup` installer during the initial product installation) is available for use during the upgrade.

The original project directory is required to upgrade your deployment. You cannot perform an upgrade without the initial project directory because it contains pertinent information about the cluster.

The original project directory contains the `apiconnect-up-v10.yml` file. An upgrade using the same project directory as the one used for installation will transfer the `encryption-secret` and all other certificates for the project to the upgraded version.

11. Block all external traffic to the management server.

Upgrading API Connect requires down-time while files are transferred and updated. All external traffic to the management server from all sources (Cloud Manager UI, API Manager UI, CLI, and REST API) must be blocked. As a result, the Cloud Manager and API Manager UIs cannot be used during the upgrade. In addition, the Developer Portal cannot be used to create, update, or delete any applications, subscriptions, memberships, or consumer organizations during the upgrade.

12. Review the [Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10](#) to ensure that any customizations to the Developer Portal are compatible with Drupal 10.

In API Connect 10.0.5.3, the Developer Portal moved from Drupal 9 to Drupal 10 (this upgrade also requires PHP 8.1). The upgrade tooling will update your Developer Portal sites; however, if you have any custom modules or themes, it is your responsibility to ensure their compatibility with Drupal 10 and PHP 8.1 before starting the upgrade.

Note: If you are upgrading to 10.0.5.3 from 10.0.1.7 or earlier, or from 10.0.5.0 or earlier, your Developer Portal sites will first be upgraded to Drupal 9.5 and then to Drupal 10. When you are planning your the upgrade to 10.0.5.3, consider the extended time frame. To shorten your upgrade window, consider removing the Developer Portal sites that you do not need.

Upgrading a two data center deployment

How to upgrade a two data center disaster recovery (DR) deployment on VMware.

To upgrade a two data center disaster recovery deployment, follow the [Upgrading to the latest release on VMware procedures](#) along with the additional procedures that are described in this topic.

Key points for two data center disaster recovery upgrade on VMware:

- Both data centers must be using the same version of API Connect, including down to the specific fix pack and interim fix level. Therefore, both data centers must be upgraded in the same maintenance window.

- Do not reboot any of the management or portal virtual machines until the end of the upgrade, even if you're instructed to do so by the `apicup subsys health-check` command during the upgrade process.
- 2DCDR is disabled on the management subsystems during the upgrade process. Keep a note of which data center had the active management subsystems when you start the upgrade process; set this same data center as the active at the end of the process.
- For the portal subsystems, 2DCDR is not disabled during upgrade, and the warm-standby data center must be upgraded first.

Upgrade the subsystems in the following order:

1. Review the [Upgrade considerations on VMware](#).
2. Complete the [Preparing to upgrade on VMware](#) steps.
3. Complete the pre-upgrade verification of your 2DCDR deployment, following the [Upgrade on VMware](#) steps, but with the following changes:
 - Take the Postgres backup (step 2) on your active management subsystem only.
 - All **apicops** operations are run on the active management subsystem only.
 - For the management subsystem, stop before you start the [Upgrade the management subsystem](#) step, and proceed to step 4 instead. Do not run:


```
apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive>
```
4. To upgrade the management subsystem, follow [Management subsystem two data center upgrade](#).
5. To upgrade the portal subsystem, follow the steps in [Upgrade on VMware](#), but ensure that the warm-standby is upgraded first.
6. Upgrade analytics and gateway subsystems as described in [Upgrade on VMware](#).

How to upgrade when one data center is down

If you cannot get your failed data center back online and want to upgrade, you can convert your active data center to a stand-alone and upgrade it:

1. Ensure that the network links to the failed data center are removed, so that if the failed data center recovers unexpectedly, it cannot interfere with your active data center.
2. Convert the active data center to a stand-alone data center: [Removing a two data center deployment](#).
3. Upgrade your active data center with the standard upgrade steps: [Upgrade on VMware](#).

To restore 2DCDR, replace the API Connect deployment in your failed data center with a new installation at the same fix pack level as your upgraded active data center. Follow the steps in [Installing a two data center deployment on VMware](#) to configure 2DCDR between the data centers.

Related concepts

- [A two data center deployment strategy on VMware](#)

Related tasks

- [Installing a two data center deployment on VMware](#)

Management subsystem two data center upgrade

How to upgrade your management subsystem in a two data center deployment.

About this task

Follow the steps in this topic, along with the steps documented in [Upgrading to the latest release on VMware](#).

The steps in this topic involve running command-line operations from several locations:

- **apicup** operations are run from your API Connect project directory.
Note: The **apicup** command is a binary file that is included with each API Connect release. Unless stated otherwise, all operations in this topic that use the **apicup** command must use the **apicup** binary from the target release (the release that you want to upgrade to). Be careful to not use the **apicup** command from your source release.
- **apic** and **kubectl** operations are run from within your management appliance, as the root user. Login to your management virtual appliance with an SSH client and the username **apicadm**, then switch to the root user with **sudo -i**.

Procedure

1. Determine the current version and license of API Connect (the version that you are upgrading from). You will need to specify the correct version number and license in the next step.
 - a. To get the current version, from within the management appliance run the following command:

```
apic version
```

In the response, the `- subsystem- <subsystem-name>` statement displays the installed version. In the following example, the response indicates that the installed version is 10.0.5.2:

```
apic version
```

```
INFO[0000] Log level: info
Control plane: 1.26.0
Appliance base 4.6.0
GitCommitSha:10.0.5.2-55-gb784ac7
Packages:
- appliance-base:4.6.0
```

- appliance-runtime-4.6.0:1.0.0
- appliance-setup-4.6.0:1.0.0
- subsystem-management:1:10.0.5.2

Important: If you are upgrading from a source version of 10.0.1.x, you must append `-eus` to the end of all references to that version number; for example `10.0.1.2-ifix2-eus`.

- To get the current license, run the following `apicup` command in your project directory:

```
apicup licenses list
```

in the response, the `ID` statement displays the installed license, for example:

```
Accepted licenses:
  ID: <source_license>
  Product Version: <source_version>
  URL: <license_url>
```

Note: If there are multiple accepted licenses, refer only to the `<source_license>` that aligns with the `<source_version>` of your system (that is the `<source_version>` that matches the version that you obtained in step 1.a).

- Add source license and version to your `extra-values.yaml` file.

Note: If you do not have an `extra-values.yaml` file, then create one in a directory that the `apicup` command can access.

Add the source lines that you identified in step 1.b to your `extra-values.yaml` as shown:

```
spec:
  version: "<source_version>"
  license:
    license: <source_license>
  imageRegistry: 127.0.0.1:8675/<source_version>
```

- Apply the `extra-values.yaml` file.

Run the following `apicup` command to set the `extra-values` file for both the `active` and `warm-standby` sites:

```
apicup subsys set <mgmt> extra-values-file=<extra-values-file-path>
```

where `<mgmt>` is the name of your management subsystem, which you can get with:

```
apicup subsys list
```

If you want to validate your `<extra-values-file-path>`, and the other subsystem settings, before proceeding, you can run:

```
apicup subsys get <mgmt> --validate
```

- Start the upgrade process on both active and warm-standby management subsystems.

```
apicup subsys install <mgmt> <mgmt_upgrade_tar_archive>
```

Note: If any Control Plane files are needed for your upgrade, you must append the files to the `install` command.

- Wait for the management subsystem status to become `Running` on both sites.

To determine the status, login to the management appliances on both sites as root user and run:

```
kubectl get mgmt
```

NAME	READY	STATUS	VERSION	RECONCILED	VERSION AGE
management	n/n	Running	10.0.5.2	10.0.5.2-100	5h2m

The `RECONCILED`

`VERSION/VERSION` should be the `source_version` that you retrieved in step 1.

- Disable 2DCDR on the `warm-standby` management subsystem:

```
apicup subsys set <warm-standby management subsystem> multi-site-ha-enabled=false
apicup subsys install <warm-standby management subsystem>
```

Conversion from warm-standby to a stand-alone management subsystem can take some time. Run the health check to ensure that the deployment is in a good state before proceeding:

```
apicup subsys health-check <warm-standby management subsystem>
```

an empty response means no detected problems.

- Disable 2DCDR on the active management subsystem.

```
apicup subsys set <active management subsystem> multi-site-ha-enabled=false
apicup subsys install <active management subsystem>
```

Conversion from active to a stand-alone management subsystem can take some time. Run the health check to ensure that the deployment is in a good state before proceeding:

```
apicup subsys health-check <active management subsystem>
```

an empty response means no detected problems.

- Wait for the management subsystem status to become `Running` on both sites.

To determine the status, login to the management appliances on both sites as root user and run:

```
kubectl get mgmt
```

NAME	READY	STATUS	VERSION	RECONCILED	VERSION AGE
management	n/n	Running	10.0.5.2	10.0.5.2-100	5h2m

The `RECONCILED`

`VERSION/VERSION` should be the `source_version` that you retrieved in step 1.

- Remove the source license and version from your `extra-values.yaml` file.

Delete the `version:` and `license:` lines that you added to your `extra-values.yaml` file in step 2. Apply this change to both sites by running:

```
apicup subsys install <mgmt>
```

10. Wait for the management subsystems status to become **Running** on both sites.

```
kubectl get mgmt
NAME      READY   STATUS    VERSION    RECONCILED VERSION AGE
management 16/16   Running  10.0.5.3   10.0.5.3-100 5h2m
```

The **RECONCILED VERSION/VERSION** should now be the target release that you are upgrading to.

11. Run the health check on both clusters to ensure that the upgrade completed.

```
apicup subsys health-check <mgmt>
```

an empty response means no detected problems.

12. Enable 2DCDR on the previously **active** management subsystem.

```
apicup subsys set <active management subsystem> multi-site-ha-enabled=true
apicup subsys install <active management subsystem>
```

13. Wait for the management subsystem status to become **Running** on the active site.

```
kubectl get mgmt
NAME      READY   STATUS    VERSION    RECONCILED VERSION AGE
management 16/16   Running  10.0.5.3   10.0.5.3-100 5h2m
```

The **RECONCILED VERSION/VERSION** should now be the target release that you are upgrading to.

14. Enable 2DCDR on the previously **warm-standby** management subsystem.

```
apicup subsys set <warm-standby management subsystem> multi-site-ha-enabled=true
apicup subsys install <warm-standby management subsystem>
```

15. Wait for the management subsystem status to become **Running** on the warm-standby site.

```
kubectl get mgmt
NAME      READY   STATUS    VERSION    RECONCILED VERSION AGE
management 16/16   Running  10.0.5.3   10.0.5.3-100 5h2m
```

The **RECONCILED VERSION/VERSION** should now be the target release that you are upgrading to.

16. Verify that 2DCDR replication is working.

To validate that your two data center deployment is replicating data, login to your active management appliance as root user, run `kubectl get mgmt`:

```
kubectl get mgmt
NAME      READY   STATUS    VERSION    RECONCILED VERSION    MESSAGE                AGE
management n/n     Running  10.0.5.3-0 10.0.5.3-100          Management is ready.   8m59s
```

If you see **HA Status Warning** in the **MESSAGE** column, then check the status details with `kubectl describe mgmt`:

```
status:
  haStatus:
    {
      "lastTransitionTime": "2023-03-31T19:47:08Z",
      "message": "Replication not working, install or upgrade in progress.",
      "reason": "na",
      "status": "True",
      "type": "Pending"
    }
}
```

Notes:

- It takes time for the **warm-standby** site to sync with the **active** site.
- When **STATUS** shows **Running**, you can reboot (if `apicup subsys health-check` advises you to do so).

Upgrading to the latest release on VMware

You can upgrade API Connect on VMware.

Before you begin

- Before starting an upgrade, review the supported upgrade paths from prior versions: [Upgrade considerations on VMware](#).
- Enable keep-alive in your SSH configuration to avoid problems during the upgrade. For information, consult the documentation for your version of SSH.

Attention: Upgrading directly to v10.0.5 from releases earlier than 10.0.1.6-ifix1-eus might run into problems. For best results, first upgrade to 10.0.1.6-ifix1-eus and make a full set of backups before you proceed to upgrade to v10.0.5.

Upgrading to v10.0.5 from v10.0.4-ifix3, or from 10.0.1.6-ifix1-eus (or higher), pay attention to the following points:

- Upgrading to v10.0.5 results in the deletion of existing analytics data. If you want to retain your analytics, data then you must export it before upgrade. For instructions on exporting, see [Additional considerations for upgrading analytics from a version prior to v10.0.5.0](#). When you upgrade subsystems in this scenario, you must include the `--accept-data-loss` on all `apicup install` commands that result in an upgrade from the older releases. Since you upgrade the management subsystem first, including the parameter at that point

commits you to the loss of data, and the operation cannot be undone.

- Delete any `template` section from your analytics CR before upgrade. See [Update the analytics extra-values file](#).

About this task

You can upgrade the Management subsystem, Developer Portal subsystem, and Analytics subsystem. The Gateway subsystem remains available during the upgrade of the other subsystems.

For the optional components API Connect Toolkit and API Connect Local Test Environment, you do not need to upgrade. For these components, install the new version of each after you upgrade the subsystems.

Important notes:

- When installing the new files, be sure to use the new `apicup` installer.
- The upgrade order for subsystems is important. Upgrade the subsystems in the following order: (1) Management, (2) Portal, (3) Analytics, (4) Gateway. Management must be upgraded first. Gateway must be upgraded after Management because upgrading the Management service before the Gateway ensures that any new policies and capabilities will be available to a previously registered Gateway service.
- When you run the install, the program sends the compressed tar file, which contains the upgrade images, to all cluster members. The compressed tar file is about 2 GB, and transfer can take some time. When the install command exits, the compressed tar file has arrived at each member. The upgrade process is now underway, and might continue for some time depending on factors such as the size of your deployment, your network speed, and so on.
- The `apicup subsys install` command automatically runs `apicup health-check` prior to attempting the upgrade. An error is displayed if a problem is found that will prevent successful upgrade. When troubleshooting a problem with an upgrade, you can optionally suppress the health check. See [Skipping health check when re-running upgrade](#).

- The certificate manager was upgraded in API Connect version 10.0.5.1. If you are upgrading from an earlier release, then after you trigger the upgrade from the `apim`, `lur`, and `taskmanager` pods will be in a CrashLoopBackoff state due to Postgres requiring a certificate refresh. Correct the problem by completing the following steps:

1. SSH into the server:

- a. Run the following command to connect as the API Connect administrator, replacing `ip_address` with the appropriate IP address:

```
ssh ip_address -l apicadm
```

- b. When prompted, select Yes to continue connecting.

- c. When you are connected, run the following command to receive the necessary permissions for working directly on the appliance:

```
sudo -i
```

2. Run the following command to delete the Postgres pods, which refreshes the new certificate:

```
kubectl get pod --no-headers=true | grep postgres | grep -v backup | awk '{print $1}' | xargs kubectl delete pod
```

Procedure

Attention: If you encounter an issue while performing the upgrade, see [Troubleshooting upgrades on VMware](#).

1. Complete the prerequisites:

- a. Ensure that your deployment meets the upgrade requirements. See [Upgrade considerations on VMware](#).

- b. Complete the steps in [Preparing to upgrade on VMware](#) to ensure that your environment is ready for the upgrade.

- c. Verify the API Connect deployment is healthy and fully operational. See [Checking cluster health on VMware](#)

- d. Remove any stale upgrade files:

```
apic clean-upgrade-files
```

- e. Verify sufficient free disk space.

For each appliance node that you are planning to upgrade:

- i. SSH into the appliance, and switch to user `root`.

- ii. Check disk space in `/data/secure`:

```
df -h /data/secure
```

Make sure the disk usage shown is below 70%. If it is not, add disk capacity. See [Adding disk space to a VMware appliance](#).

- iii. Check free disk space in `/`:

```
df -h /
```

Make sure usage is below 70%. If it is not, consider deleting or offloading older `/var/log/syslog*` files.

- f. Complete a manual backup of the API Connect subsystems as explained in [Backing up and restoring on VMware](#).

Notes:

- Do not start an upgrade if a backup is scheduled to run within a few hours.
- Do not perform maintenance tasks such as rotating key-certificates, restoring from a backup, or starting a new backup, at any time while the upgrade process is running.

2. Back up the Postgres images on one of the Management Server VMs.

Follow the steps for the version of API Connect that you are upgrading from:

- 10.0.1.9 or later, and 10.0.5.1 or later:

```
SSH to one of the Management Server VMs:  
ssh <ip_address of management vm> -l apicadm  
sudo to root:  
sudo -i
```


Run the following commands:

```
postgres_operator=$(kubectl get pod -l app.kubernetes.io/name=postgres-operator -o name)

ctr --namespace k8s.io images pull --plain-http=true `kubectl get $postgres_operator -o
jsonpath='{.spec.containers[].env[?(@.name=="RELATED_IMAGE_PGO_RMDATA")].value}'`
ctr --namespace k8s.io images export rmdata.tar `kubectl get $postgres_operator -o
jsonpath='{.spec.containers[].env[?(@.name=="RELATED_IMAGE_PGO_RMDATA")].value}'`
ctr --namespace k8s.io images export backrestrepo.tar `kubectl get $postgres_operator -o
jsonpath='{.spec.containers[].env[?(@.name=="RELATED_IMAGE_PGO_BACKREST_REPO")].value}'`
ctr --namespace k8s.io images export pgbouncer.tar `kubectl get $postgres_operator -o
jsonpath='{.spec.containers[].env[?(@.name=="RELATED_IMAGE_CRUNCHY_PGBOUNCER")].value}'`
ctr --namespace k8s.io images export postgres-ha.tar `kubectl get $postgres_operator -o
jsonpath='{.spec.containers[].env[?(@.name=="RELATED_IMAGE_CRUNCHY_POSTGRES_HA")].value}'`

postgres_pod=$(kubectl get pod -l role=master -o name)
ctr --namespace k8s.io images export k8s-init.tar `kubectl get $postgres_pod -o
jsonpath='{.spec.initContainers[].image}'`
```

- 10.0.1.8 or earlier, and 10.0.4.0-ifix3 or earlier:

```
SSH to one of the Management Server VMs:
ssh <ip_address of management vm> -l apicadm
Sudo to root:
sudo -i
```

Run the following commands:

```
postgres_operator=$(kubectl get pod -l app.kubernetes.io/name=postgres-operator -o name)

docker pull `kubectl get $postgres_operator -o jsonpath='{.spec.containers[].env[?
(@.name=="RELATED_IMAGE_PGO_RMDATA")].value}'`
docker pull `kubectl get $postgres_operator -o jsonpath='{.spec.containers[].env[?
(@.name=="RELATED_IMAGE_PGO_BACKREST_REPO")].value}'`
docker pull `kubectl get $postgres_operator -o jsonpath='{.spec.containers[].env[?
(@.name=="RELATED_IMAGE_CRUNCHY_PGBOUNCER")].value}'`
docker pull `kubectl get $postgres_operator -o jsonpath='{.spec.containers[].env[?
(@.name=="RELATED_IMAGE_CRUNCHY_POSTGRES_HA")].value}'`
docker save `kubectl get $postgres_operator -o jsonpath='{.spec.containers[].env[?
(@.name=="RELATED_IMAGE_PGO_RMDATA")].value}'` -o rmdata.tar
docker save `kubectl get $postgres_operator -o jsonpath='{.spec.containers[].env[?
(@.name=="RELATED_IMAGE_PGO_BACKREST_REPO")].value}'` -o backrestrepo.tar
docker save `kubectl get $postgres_operator -o jsonpath='{.spec.containers[].env[?
(@.name=="RELATED_IMAGE_CRUNCHY_PGBOUNCER")].value}'` -o pgbouncer.tar
docker save `kubectl get $postgres_operator -o jsonpath='{.spec.containers[].env[?
(@.name=="RELATED_IMAGE_CRUNCHY_POSTGRES_HA")].value}'` -o postgres-ha.tar

postgres_pod=$(kubectl get pod -l role=master -o name)
docker save `kubectl get $postgres_pod -o jsonpath='{.spec.initContainers[].image}'` -o k8s-init.tar
```

Note: After performing the upgrade, if you see that the Postgres pods are in the `imagePullBackOff` state, correct the issue by following steps 2 and 3 in [Management subsystem upgrade imagePullBackOff issue](#) to import each of the images you saved in this step.

3. Run the pre-upgrade health check:

- a. Run the health check on the Management subsystem:

- i. SSH into the server:

1. Run the following command to connect as the API Connect administrator, replacing `ip_address` with the appropriate IP address:

```
ssh ip_address -l apicadm
```

2. When prompted, select Yes to continue connecting.

3. When you are connected, run the following command to receive the necessary permissions for working directly on the appliance:

```
sudo -i
```

- ii. Verify that the `apicops` utility is installed by running the following command to check the current version of the utility:

```
apicops --version
```

If the response indicates that `apicops` is not available, install it now. See [The API Connect operations tool: apicops](#) in the API Connect documentation.

- iii. Run the version-check script and verify that there are no errors:

```
apicops version:pre-upgrade
```

- iv. Run the appliance health check and verify that there are no errors:

```
apicops appliance-checks:appliance-pre-upgrade
```

- b. Run the health check on the Portal subsystem:

- i. SSH into the server.

- ii. Verify that the `apicops` utility is installed by running the following command to check the current version of the utility:

```
apicops --version
```

If the response indicates that `apicops` is not available, install it now. See [The API Connect operations tool: apicops](#) in the API Connect documentation.

- iii. Run the following command to check the system status, and verify that there are no errors:

```
apicops upgrade:check-subsystem-status
```

- iv. Run the appliance health check and verify that there are no errors:

```
apicops appliance-checks:appliance-pre-upgrade
```

c. Run the health check on the Analytics subsystem:

- i. SSH into the server.
- ii. Verify that the `apicops` utility is installed by running the following command to check the current version of the utility:

```
apicops --version
```

If the response indicates that `apicops` is not available, install it now. See [The API Connect operations tool: apicops](#) in the API Connect documentation.

- iii. Run the following command to check the system status, and verify that there are no errors:

```
apicops upgrade:check-subsystem-status
```

- iv. Run the appliance health check and verify that there are no errors:

```
apicops appliance-checks:appliance-pre-upgrade
```

4. Optional: Take a Virtual Machine (VM) snapshot of all your VMs; see [Using VM snapshots for infrastructure backup and disaster recovery](#) for details. This action does require a brief outage while all of the VMs in the subsystem cluster are shut down - do not take snapshots of running VMs, as they might not restore successfully. VM snapshots can offer a faster recovery when compared to redeploying OVA's and restoring from normal backups.

Important: VM snapshots are not an alternative to the standard API Connect backups that are described in the previous steps. You must complete the API Connect backups in order to use the API Connect restore feature.

5. Obtain the API Connect files:

You can access the latest files from [IBM Fix Central](#) by searching for the API Connect product and your installed version.

The following files are used during upgrade on VMware:

IBM API Connect <version> Management Upgrade File for VMware
Management subsystem files for upgrade
IBM API Connect <version> Developer Portal Upgrade File for VMware
Developer Portal subsystem files for upgrade
IBM API Connect <version> Analytics Upgrade File for VMware
Analytics subsystem files for upgrade
IBM API Connect <version> Install Assist for <operating_system_type>
The apicup installation utility. Required for all installations on VMware.

The following files are not used during upgrade, but can be installed as new installations used to replace prior versions

IBM API Connect <version> Toolkit for <operating_system_type>
Toolkit command line utility. Packaged standalone, or with API Designer or Loopback:

- IBM API Connect <version> Toolkit for <operating_system_type>
- IBM API Connect <version> Toolkit with Loopback for <operating_system_type>
- IBM API Connect <version> Toolkit Designer with Loopback for <operating_system_type>

Not required during initial installation. After installation, you can download directly from the Cloud Manager UI and API Manager UI. See [Installing the toolkit](#).

IBM API Connect <version> Local Test Environment
Optional test environment. See [Testing an API with the Local Test Environment](#)
IBM API Connect <version> Security Signature Bundle File
Checksum files that you can use to verify the integrity of your downloads.

6. If necessary, download from the same Fix Pack page any Control Plane files that are needed. Control Plane files provide support for specific Kubernetes versions. The IBM API Connect <version> Management Upgrade File for VMware file contains the latest Control Plane file. An upgrade from the most recent API Connect version to the current version does not need a separate Control Plane file. However, when upgrading from older versions of API Connect, you must install one or more control plane files to ensure that all current Kubernetes versions are supported.

Consult the following table to see if your deployment needs one or more separate Control Plane files.

Table 1. Control Plane files needed for upgrade

Version to upgrade from	Instructions for upgrading to 10.0.5.4
<ul style="list-style-type: none">• 10.0.5.3• 10.0.1.12-eus	No control-plane file needed.

For information on the Control Plane files used with older releases, see [Control Plane files for earlier releases](#).

7. Verify that the subsystem upgrade files are not corrupted.

a. Run the following command separately for the Management, the Developer Portal, and the Analytics upgrade files:

- Mac or Linux:

```
sha256sum <upgrade-file-name>
```

Example:

```
sha256sum upgrade_management_v10.0.5.4
```

- Windows:

```
C:\> certUtil -hashfile C:\<upgrade-file-name> SHA256
```

Example:

```
C:\> certUtil -hashfile C:\upgrade_management_v10.0.5.4 SHA256
```

b. Compare the result with the checksum values to verify that the files are not corrupted.

If the checksum does not match the value for the corresponding version and subsystem, then the upgrade file is corrupted. Delete it and download a new copy of the upgrade file. The following list shows the checksum values for the current release.

Checksum values for 10.0.5.4:

```
10.0.5.4 analytics : 3a931b046799ff0ddffb6aa324b4e58c7eba6cae9bf9fdc1b878c402848a3014
10.0.5.4 management : 0b4f02abf2c10862a88af18eee98944a20fd5cb7b50e43e202ab20192ba4a68d
10.0.5.4 portal : 58ca29622dfddc62e236a2b9a9023630feb11a8ab575fd1395dd2d9a776934a9
```

For information on the checksum files used with older releases, see [Checksum values for earlier releases](#).

8. Install the installation utility.

- Locate the apicup installation utility file for your operating system, and place it in your project directory
- Rename the file for your OS type to **apicup**. Note that the instructions in this documentation refer to **apicup**.
- OSX and Linux® only: Make the apicup file an executable file by entering **chmod +x apicup**.
- Set your path to the location of your apicup file.

OSX and Linux	export PATH=\$PATH:/Users/your_path/
Windows	set PATH=c:\your_path;%PATH%

- From within your project directory, specify the API Connect license:

```
apicup licenses accept <License_ID>
```

The **<License_ID>** is specific to the API Connect Program Name you purchased. To view the supported license IDs, see [API Connect licenses](#).

9. **If upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher):** Disassociate and delete your Analytics services.

Important: Upgrading to v10.0.5.x from v10.0.4-ifix3, or from v10.0.1.6-eus (or higher) will result in the deletion of existing analytics data. If you want to retain your analytics data then you must export it before upgrade. For instructions on exporting see [Additional considerations for upgrading analytics from a version prior to v10.0.5.0](#).

- In Cloud Manager UI, click Topology.
- In the section for the Availability Zone that contains the Analytics service, locate the Gateway service that the Analytics service is associated with.
- Click the actions menu, and select Unassociate analytics service.
Remember to disassociate each Analytics service from all Gateways.
- In the section for the Availability Zone that contains the Analytics services, locate each Analytics service and click Delete.

10. Upgrade the Management subsystem:

- Run the upgrade on the Management subsystem:
 - Install the Management subsystem files.

```
apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive>
```

Notes:

- If you are adding one or more control plane files, specify each path and file name on the command line:

```
apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive> <path_to_control_plane_file_1> <path_to_control_plane_file_n>
```

- If you are upgrading to v10.0.5.x from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher), add the argument **--accept-data-loss** to the command to indicate that you accept the loss of the Analytics data:

```
apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive> --accept-data-loss
```

Remember, when you include this parameter for the Management subsystem upgrade, you commit to the loss of data and the operation cannot be undone.

- Verify that the upgrade was successful by running a health check with the following command:

```
apicup subsys health-check <subsystem_name>
```

- If the health check indicates that a reboot is needed, run the following commands:

- apic lock**
- systemctl reboot**

- Use **apicops** to validate the certificates.

For information on the **apicops** tool, see [The API Connect operations tool: apicops](#).

- Run the following command:

```
apicops upgrade:stale-certs -n <namespace>
```

- Delete any stale certificates that are managed by cert-manager.

If a certificate failed the validation and it is managed by cert-manager, you can delete the stale certificate secret, and let cert-manager regenerate it. Run the following command:

```
kubectl delete secret <stale-secret> -n <namespace>
```

- Restart the corresponding pod so that it can pick up the new secret.

To determine which pod to restart, see the following topics:

- [List of ingress certificates](#)
- [List of intra-subsystem certificates](#)

- Restart all nats-server pods.

- Run the following command to connect as the API Connect administrator, replacing **<ip_address>** with the appropriate IP address:

```
ssh <ip_address> -l apicadm
```

- When prompted, select Yes to continue connecting.

- When you are connected, run the following command to receive the necessary permissions for working directly on the appliance:

```
sudo -i
```

iv. Restart all nats-server pods by running the following command:

```
kubectl -n <namespace> delete po -l app.kubernetes.io/name=natscluster
```

11. If there are multiple Management subsystems in the same project, set the Portal subsystem's **platform-api** and **consumer-api** certificates to match those used by the appropriate Management subsystem to ensure that the Portal subsystem is correctly associated with that Management subsystem. This step only applies if you installed more than one Management subsystem into a single project.

A Portal subsystem can be associated with only one Management subsystem. To associate the upgraded Portal subsystem with the appropriate Management subsystem, manually set the **mgmt-platform-api** and **mgmt-consumer-api** certificates to match the ones used by the Management subsystem.

a. Run the following commands to get the certificates from the Management subsystem:

```
apicup certs get <mgmt-subsystem-name> platform-api -t cert > platform-api.crt
apicup certs get <mgmt-subsystem-name> platform-api -t key > platform-api.key
apicup certs get <mgmt-subsystem-name> platform-api -t ca > platform-api-ca.crt

apicup certs get <mgmt-subsystem-name> consumer-api -t cert > consumer-api.crt
apicup certs get <mgmt-subsystem-name> consumer-api -t key > consumer-api.key
apicup certs get <mgmt-subsystem-name> consumer-api -t ca > consumer-api-ca.crt
```

where **mgmt-subsystem-name** is the name of the specific Management subsystem that you want to associate the new Portal subsystem with.

b. Run the following commands to set the Portal's certificates to match those used by the Management subsystem:

```
apicup certs set <ptl-subsystem-name> platform-api Cert_file_path Key_file_path CA_file_path
apicup certs set <ptl-subsystem-name> consumer-api Cert_file_path Key_file_path CA_file_path
```

For more information on **apicup** certificate commands, see [Command reference](#).

12. Upgrade the Portal subsystem.

a. If you did not verify that your Portal customizations are compatible with Drupal 10, do that now.

In API Connect 10.0.5.3, the Developer Portal moved from Drupal 9 to Drupal 10 (this upgrade also requires PHP 8.1). The upgrade tooling will update your Developer Portal sites; however, if you have any custom modules or themes, it is your responsibility to ensure their compatibility with Drupal 10 and PHP 8.1 before starting the upgrade. Review the [Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10](#) to ensure that any customizations to the Developer Portal are compatible with Drupal 10 and PHP 8.1.

b. Install the Portal subsystem files:

```
apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive>
```

Notes:

- If you are adding one or more control plane files, specify each path and file name on the command line:

```
apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive> <path_to_control_plane_file_1>
<path_to_control_plane_file_n>
```

- If you are upgrading to v10.0.5.x from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher), add the argument **--accept-data-loss** to the command to indicate that you accept the loss of the Analytics data:

```
apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive> --accept-data-loss
```

c. Verify that the upgrade was successful by running a health check with the following command:

```
apicup subsys health-check <subsystem_name>
```

Important: If the health check indicates that a reboot is required, first complete the upgrade for all Portal sites and verify that all sites were upgraded before running the reboot command at the end of this step.

d. Run the following toolkit commands to ensure that the Portal site upgrades are complete:

i. Log in as an admin user:

```
apic login -s <server_name> --realm admin/default-idp-1 --username admin --password <password>
```

ii. Get the portal service ID and endpoint:

```
apic portal-services:get -o admin -s <management_server_endpoint> \
--availability-zone availability-zone-default <portal-service-name> \
--output - --format json
```

iii. List the sites:

```
apic --mode portaladmin sites:list -s <management_server_endpoint> \
--portal_service_name <portal-service-name> \
--format json
```

Any sites currently upgrading display the **UPGRADING** status; any site that completed its upgrade displays the **INSTALLED** status and the new platform version. Verify that all sites display the **INSTALLED** status before proceeding.

For more information on the **sites** command, see [apic sites:list](#) and [Using the sites commands](#).

iv. After all sites are in **INSTALLED** state and have the new platform listed, run:

```
apic --mode portaladmin platforms:list -s <server_name> --portal_service_name <portal_service_name>
```

Verify that the new version of the platform is the only platform listed.

For more information on the **platforms** command, see [apic platforms:list](#) and [Using the platforms commands](#).

e. If all Portal sites are upgraded and the health check indicates that a reboot is needed, then run the following command:

```
systemctl reboot
```

13. Upgrade the Analytics subsystem

Important: Upgrading to v10.0.5.x from v10.0.4-ifix3, or from v10.0.1.6-eus (or higher) will result in the deletion of existing analytics data. If you want to retain your analytics data then you must export it before upgrade. For instructions on exporting see [Additional considerations for upgrading analytics from a version prior to v10.0.5.0](#).

a. Install the Analytics subsystem files:

```
apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive>
```

Notes:

- If you are adding one or more control plane files, specify each path and file name on the command line:

```
apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive> <path_to_control_plane_file_1> <path_to_control_plane_file_n>
```

- If you are upgrading to v10.0.5.x from v10.0.4-ifix3, or upgrading from v10.0.1.7-eus (or higher), add the argument `--accept-data-loss` to the command to indicate that you accept the loss of the Analytics data:

```
apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive> --accept-data-loss
```

b. Verify that the upgrade was successful by running a health check with the following command:

```
apicup subsys health-check <subsystem_name>
```

c. If the health check indicates that a reboot is needed, run the following commands:

i. `apic lock`

ii. `systemctl reboot`

14. If you are upgrading from v10.0.4-ifix3, or upgrading from v10.0.1.6-eus (or higher): Enable the analytics service as explained in [Enabling Analytics after upgrading](#).

15. If you are upgrading from a release earlier than v10.0.5.3: If you want to use JWT security instead of mTLS, enable this feature as explained in [Use JWT security instead of mTLS between subsystems](#).

16. Complete a manual backup of the upgraded API Connect subsystems; see [Backing up and restoring on VMware](#).

17. Optional: Take a Virtual Machine (VM) snapshot of all your VMs; see [Using VM snapshots for infrastructure backup and disaster recovery](#) for details. This action requires a brief outage while all of the VMs in the subsystem cluster are shut down - do not take snapshots of running VMs, as they might not restore successfully.

18. Complete the disaster preparation steps to ensure recovery of API Connect from a disaster event. See [Preparing for a disaster](#).

What to do next

When you have upgraded all of the API Connect subsystems, upgrade your DataPower Gateway Service. See [Upgrading DataPower Gateway Service](#).

- [Control Plane files for earlier releases](#)

If you are upgrading to an earlier version than 10.0.5.4, refer to the following tables to determine which Control Plane files are required.

- [Checksum values for earlier releases](#)

If you are upgrading to an earlier version than 10.0.5.4, use the corresponding checksum values to verify your downloaded upgrade files.

Control Plane files for earlier releases

If you are upgrading to an earlier version than 10.0.5.4, refer to the following tables to determine which Control Plane files are required.

Attention: Upgrading directly to the latest release from releases earlier than 10.0.1.6-ifix1-eus might run into problems. For best results, first upgrade to 10.0.1.6-ifix1eus and make a full set of backups before proceeding to upgrade to the latest release.

Control planes for upgrading to 10.0.5.4

Installs with `appliance-control-plane-1.26.x.tgz`

Table 1. Control Plane files needed for upgrading to 10.0.5.4

API Connect version	Control Plane files needed
<ul style="list-style-type: none">• 10.0.5.3• 10.0.1.12-eus	No control-plane file needed.
<ul style="list-style-type: none">• 10.0.5.2• 10.0.1.11-eus• 10.0.1.9-eus	Download: <ul style="list-style-type: none">• <code>appliance-control-plane-1.25.x.tgz</code>
<ul style="list-style-type: none">• 10.0.5.1• 10.0.1.8-eus• 10.0.1.7-eus	Download: <ul style="list-style-type: none">• <code>appliance-control-plane-1.25.x.tgz</code>• <code>appliance-control-plane-1.24.x.tgz</code>
<ul style="list-style-type: none">• 10.0.4.0 and iFixes	Download: <ul style="list-style-type: none">• <code>appliance-control-plane-1.25.x.tgz</code>• <code>appliance-control-plane-1.24.x.tgz</code>• <code>appliance-control-plane-1.23.x.tgz</code>

API Connect version	Control Plane files needed
<ul style="list-style-type: none"> 10.0.3.0 and iFixes 10.0.1.6-eus and iFixes 10.0.1.5-eus and iFixes 10.0.1.4-eus and iFixes 	Download: <ul style="list-style-type: none"> appliance-control-plane-1.25.x.tgz appliance-control-plane-1.24.x.tgz appliance-control-plane-1.23.x.tgz appliance-control-plane-1.22.x.tgz
<ul style="list-style-type: none"> 10.0.2.0 10.0.1.2-eus and iFixes 	Download: <ul style="list-style-type: none"> appliance-control-plane-1.25.x.tgz appliance-control-plane-1.24.x.tgz appliance-control-plane-1.23.x.tgz appliance-control-plane-1.22.x.tgz appliance-control-plane-1.21.x.tgz

Control planes for upgrading to 10.0.5.3

Installs with `appliance-control-plane-1.26.x.tgz`

Table 2. Control Plane files needed for upgrading to 10.0.5.3

API Connect version	Control Plane files needed
<ul style="list-style-type: none"> 10.0.5.2 10.0.1.11-eus 10.0.1.9-eus 	Download: <ul style="list-style-type: none"> appliance-control-plane-1.25.x.tgz
<ul style="list-style-type: none"> 10.0.5.1 10.0.1.8-eus 10.0.1.7-eus 	Download: <ul style="list-style-type: none"> appliance-control-plane-1.25.x.tgz appliance-control-plane-1.24.x.tgz
<ul style="list-style-type: none"> 10.0.4.0 and iFixes 	Download: <ul style="list-style-type: none"> appliance-control-plane-1.25.x.tgz appliance-control-plane-1.24.x.tgz appliance-control-plane-1.23.x.tgz
<ul style="list-style-type: none"> 10.0.3.0 and iFixes 10.0.1.6-eus and iFixes 10.0.1.5-eus and iFixes 10.0.1.4-eus and iFixes 	Download: <ul style="list-style-type: none"> appliance-control-plane-1.25.x.tgz appliance-control-plane-1.24.x.tgz appliance-control-plane-1.23.x.tgz appliance-control-plane-1.22.x.tgz
<ul style="list-style-type: none"> 10.0.2.0 10.0.1.2-eus and iFixes 	Download: <ul style="list-style-type: none"> appliance-control-plane-1.25.x.tgz appliance-control-plane-1.24.x.tgz appliance-control-plane-1.23.x.tgz appliance-control-plane-1.22.x.tgz appliance-control-plane-1.21.x.tgz

Control planes for upgrading to 10.0.1.11-eus

Installs with `appliance-control-plane-1.24.x.tgz`

Table 3. Control Plane files needed for upgrading to 10.0.1.11-eus

API Connect version	Control Plane files needed
<ul style="list-style-type: none"> 10.0.5.1 10.0.1.9-eus 10.0.1.8-eus 10.0.1.7-eus 	No Control Plane file needed.
<ul style="list-style-type: none"> 10.0.4.0 and iFixes 	Download: <ul style="list-style-type: none"> appliance-control-plane-1.23.x.tgz
<ul style="list-style-type: none"> 10.0.3.0 and iFixes 10.0.1.6-eus and iFixes 10.0.1.5-eus and iFixes 10.0.1.4-eus and iFixes 	Download: <ul style="list-style-type: none"> appliance-control-plane-1.23.x.tgz appliance-control-plane-1.22.x.tgz
<ul style="list-style-type: none"> 10.0.2.0 10.0.1.2-eus and iFixes 	Download: <ul style="list-style-type: none"> appliance-control-plane-1.23.x.tgz appliance-control-plane-1.22.x.tgz appliance-control-plane-1.21.x.tgz

Control planes for upgrading to 10.0.5.2

Installs with `appliance-control-plane-1.24.x.tgz`

Table 4. Control Plane files needed for upgrading to 10.0.5.2

API Connect version	Control Plane files needed
<ul style="list-style-type: none">• 10.0.5.1• 10.0.1.9-eus• 10.0.1.8-eus• 10.0.1.7-eus	No Control Plane file needed.
<ul style="list-style-type: none">• 10.0.4.0 and iFixes	Download: <ul style="list-style-type: none">• <code>appliance-control-plane-1.23.x.tgz</code>
<ul style="list-style-type: none">• 10.0.3.0 and iFixes• 10.0.1.6-eus and iFixes• 10.0.1.5-eus and iFixes• 10.0.1.4-eus and iFixes	Download: <ul style="list-style-type: none">• <code>appliance-control-plane-1.23.x.tgz</code>• <code>appliance-control-plane-1.22.x.tgz</code>
<ul style="list-style-type: none">• 10.0.2.0• 10.0.1.2-eus and iFixes	Download: <ul style="list-style-type: none">• <code>appliance-control-plane-1.23.x.tgz</code>• <code>appliance-control-plane-1.22.x.tgz</code>• <code>appliance-control-plane-1.21.x.tgz</code>

Control planes for upgrading to 10.0.1.9-eus

Installs with `appliance-control-plane-1.24.x.tgz`

Table 5. Control Plane files needed for earlier releases

API Connect version	Control Plane files needed
<ul style="list-style-type: none">• 10.0.1.8-eus• 10.0.1.7-eus	No Control Plane file needed.
<ul style="list-style-type: none">• 10.0.4.0 and iFixes	Download: <ul style="list-style-type: none">• <code>appliance-control-plane-1.23.x.tgz</code>
<ul style="list-style-type: none">• 10.0.3.0 and iFixes• 10.0.2.0 and iFixes• 10.0.1.6-eus and iFixes• 10.0.1.5-eus and iFixes• 10.0.1.4-eus and iFixes• 10.0.1.2-eus and iFixes	Download: <ul style="list-style-type: none">• <code>appliance-control-plane-1.23.x.tgz</code>• <code>appliance-control-plane-1.22.x.tgz</code>

Control planes for upgrading to 10.0.5.1

Installs with `appliance-control-plane-1.23.x.tgz`

Note: Version 10.0.5.0 does not support VMware.

Table 6. Control Plane files needed for earlier releases

API Connect version	Control Plane files needed
<ul style="list-style-type: none">• 10.0.1.8-eus• 10.0.1.7-eus	No Control Plane file needed.
<ul style="list-style-type: none">• 10.0.4.0 and iFixes	No Control Plane file needed.
<ul style="list-style-type: none">• 10.0.3.0 and iFixes• 10.0.2.0 and iFixes• 10.0.1.6-eus and iFixes• 10.0.1.5-eus and iFixes• 10.0.1.4-eus and iFixes• 10.0.1.2-eus and iFixes	Download: <ul style="list-style-type: none">• <code>appliance-control-plane-1.22.x.tgz</code>

Control planes for upgrading to 10.0.1.7-eus and 10.0.1.8-eus

Installs with `appliance-control-plane-1.23.x.tgz`

Table 7. Control Plane files needed for 10.0.1.7-eus and 10.0.1.8-eus

API Connect version	Control Plane files needed
<ul style="list-style-type: none">• 10.0.4.0 and iFixes	No Control Plane file needed.

API Connect version	Control Plane files needed
<ul style="list-style-type: none"> 10.0.3.0 and iFixes 10.0.2.0 and iFixes 10.0.1.6-eus and iFixes 10.0.1.5-eus and iFixes 10.0.1.4-eus and iFixes 10.0.1.2-eus and iFixes 	Download: <ul style="list-style-type: none"> <code>appliance-control-plane-1.22.x.tgz</code>

Control planes for upgrading to 10.0.4.0

Installs with `appliance-control-plane-1.22.x.tgz`

Table 8. Control Plane files needed for 10.0.4.0

API Connect version	Control Plane files needed
<ul style="list-style-type: none"> 10.0.3.0 and iFixes 10.0.2.0 and iFixes 10.0.1.6-eus and iFixes 10.0.1.5-eus and iFixes 10.0.1.4-eus and iFixes 10.0.1.2-eus and iFixes 	No Control Plane file needed.

Checksum values for earlier releases

If you are upgrading to an earlier version than 10.0.5.4, use the corresponding checksum values to verify your downloaded upgrade files.

Attention: Upgrading directly to the latest release from releases earlier than 10.0.1.6-ifix1-eus might run into problems. For best results, first upgrade to 10.0.1.6-ifix1eus and make a full set of backups before proceeding to upgrade to the latest release.

Version 10.0.5.0 did not support VMware so there are no checksum values for that release.

```
10.0.5.4 analytics : 3a931b046799ff0ddff6aa324b4e58c7eba6cae9bf9fdc1b878c402848a3014
10.0.5.4 management : 0b4f02abf2c10862a88af18eee98944a20fd5cb7b50e43e202ab20192ba4a68d
10.0.5.4 portal : 58ca29622dfddc62e236a2b9a9023630feb11a8ab575fd1395dd2d9a776934a9
```

```
10.0.5.3 analytics : 94d73a54254580182cee045f551c72a019c4bfb225710fd22606a74fb890b9f9
10.0.5.3 management : 24cd8ea00b06621fcc9f26b381fdc807feafad42a4e4971ed9ac7bee6fc3e371
10.0.5.3 portal : d44180c4c417f3b39c7f5cbc695a2998ee42b987deda59019d7b0e1eb13a25e3
```

```
10.0.5.2 analytics : 0d3250db4ed30aac68d1b44db4236498a6526c6cac6688f0b5b77f78d348db35
10.0.5.2 management : 00c37d782116d5f985882b3a3a9b3312dbc5f91b57c6ae84c65a5f6af32c910c
10.0.5.2 portal : b3eff8b07c981c46cbcf03462abadb714dcb5122a1e97e95fba64f4b919c79ce
```

```
v10.0.5.1 analytics : b09cf458e7e0445854b1eb5f1b8dbbb04abc290c25e410e8cebbaf31e9275b6b
v10.0.5.1 management : 167f36158a468b0dca07ae899ac2baa6379f13dbefaad87ca8cb51142afcb22
v10.0.5.1 portal : cdaf0701eb51ecfb7242f7ad50eab3a9207601807a15d3ffbcb10e7938654a0
```

```
v10.0.1.9 analytics : fd4b3bcab78dc251d2deb3a33294021d126646dedba6a27cb4fa4ea0013ad632
v10.0.1.9 management : d2cf7dd2cab0feb1b74491ea8c56061fd388eab77eb498a17011e71ba5f770bc
v10.0.1.9 portal : 929a6a1ae54a2b1ae498790708936fd71505edae2958fc0987c55852ead65fe
```

```
v10.0.1.8 analytics : 4e389610fff67709de0968d94d1d5d0d8bae895d1f87b4386413a91cd1ddf7ae
v10.0.1.8 management : 44c05690ee4a4a94c5cd3b22a13f6b1af55f9c148eba92527d2b7f1e91da7bc5
v10.0.1.8 portal : 54dd15cf39694f8d45c9ec21a4f7b2e367b365a8a90d49bf4e13ca85f00bd888
```

```
v10.0.1.7 analytics : cf88698c5ee1d73e1dd213ab1533c3949bb9456f92580d658a806fbd8bd85e27
v10.0.1.7 management : a5c7244fbbaeafeafe9f4be0b8dfb3f9243b60326b264b85489d4a9d7690dceb
v10.0.1.7 portal : db18a212a7449c7bedb7d2369db9a23cd6424e74f21fd6125efa9d163b2381b
```

```
v10.0.4.0-ifix3 analytics : db742d6c01a8d2f147c7c455a93395d92cc97255b847f5699fff07cd9808fd19
v10.0.4.0-ifix3 management : 8f23cd4f295362ec64efcc29d0a7c9d67f1f09d3893b6707b0b3075c88f54a6a
v10.0.4.0-ifix3 portal : 7f5e29e5c1f3fb17e4b719d421a02b50d9052e3b9afefeb66450bdb8a773c2b48
```

Upgrading DataPower Gateway Service

Use these instructions when upgrading DataPower Gateway Service in a non-Kubernetes environment, during upgrade of API Connect.

Before you begin

Ensure you have completed the upgrade of the API Connect subsystems (Management service, Developer Portal, Analytics) before upgrading DataPower Gateway. Upgrading the Management service before the DataPower Gateway ensures that any new policies and capabilities will be available to a previously registered Gateway service.

About this task

- The DataPower Gateway Service in a non-Kubernetes environment can be on either a physical appliance or in a virtual DataPower deployment.
- The upgrade of DataPower Gateway Service in a non-Kubernetes environment, for use in API Connect deployments in a VMware environment, does not use the `apicup` command. Use DataPower Gateway instructions in the following steps.

Note: To upgrade the DataPower Gateway Service in a Kubernetes environment, do not use this procedure. Instead, see [Upgrading Gateway subsystem on native Kubernetes](#).

Procedure

1. Ensure that DataPower Gateway firmware version you plan to install is compatible with the API Connect Management server version.

Note:

You can use any combination of API Connect 10.0.5.x with DataPower Gateway 10.5.0.x or DataPower API Gateway 10.5.0.x.

Before you install, best practice is to review the latest compatibility support for your version of API Connect. To view compatibility support, follow the instructions in [IBM API Connect Version 10 software product compatibility requirements](#) to access API Connect information on the Software Product Compatibility Reports website. Once you access information for your version of API Connect, select Supported Software, Integration Middleware, and view the list of compatible DataPower Gateway versions.

2. When upgrading a high-availability cluster, ensure that you meet the requirements:

- Gateways must be updated one at a time.
- Before starting the upgrade, a single gateway must be running as primary for all gateway-peering definitions.
- When upgrading multiple gateways, the primary gateway must be upgraded last.

To determine which gateway is running as primary, use either the `show`

`gateway-peering-status` command in the DataPower CLI, or use the Gateway Peering Status display in the Web GUI in the API Connect application domain. To move the primary to the DataPower on which you're currently working, you can issue the `gateway-peering-switch-primary <peering-object-name>` command.

3. Ensure that your deployment includes an NTP server to synchronize time between each of the DataPower Gateways.

See [Managing the NTP service](#).

4. Ensure that you have set a unique `System Identifier` for each v10 DataPower gateway. See [Initializing the DataPower Gateway](#).

5. Follow the upgrade instructions on the DataPower Gateway documentation. See [Installation operations](#).

Note:

- If upgrading a cluster (high-availability) deployment, ensure that you have identified which gateway is running a primary in gateway-peering definitions, and that you upgrade that gateway last.
- For troubleshooting help with DataPower operator deployments, see [IBM DataPower Operator documentation](#).

Upgrading from 2018 on VMware

You can upgrade API Connect on VMware from the latest release of v2018 to the latest release of v10.

Complete the API Connect upgrade from v2018 to v10 by performing the following tasks:

- [Upgrade from v2018: overview and considerations](#)
Review the upgrade overview, requirements, and limitations before upgrading your API Connect 2018 deployment to v10 on VMware.
- [Preparing the 2018 deployment for upgrade](#)
Prepare your 2018 deployment for upgrade to the latest v10 on VMware.
- [Upgrading v2018 subsystems to v10](#)
Install the new v10 system and configure each subsystem with configuration information from 2018.

Upgrade from v2018: overview and considerations

Review the upgrade overview, requirements, and limitations before upgrading your API Connect 2018 deployment to v10 on VMware.

- [Overview of upgrade procedure](#)
- [Planning a v10 deployment](#)
- [Required backups of 2018 subsystems](#)
- [Required disk space for extraction of 2018 management subsystem data](#)
- [Automatic shortening of subsystem names with greater than 63 characters](#)
- [Upgrade Limitations](#)
- [Getting started with the upgrade procedure](#)

Overview of upgrade procedure

The upgrade from v2018 to the latest release of v10 is a fresh install of the v10 subsystems, followed by restoration of saved 2018 data and configuration.

For the management subsystem, you will use custom scripts to extract management database configuration from 2018, and then load it onto a fresh install of v10. The scripts complete the necessary conversion of the Management database information from Cassandra format to Postgres format. The scripts are provided as part of the latest API Connect 10 distribution files on Fix Central.

For the Developer Portal, you will perform a fresh install of the subsystem and then restore data from the 2018 backups.

For Analytics, you will perform a fresh install of the subsystem but you cannot restore data from the 2018 backups due to changes in the v10 Analytics subsystem.

When all subsystem upgrades are complete, you can upgrade DataPower Gateway.

Gateways can continue to run during this process so API traffic is not impacted.

Summary of the upgrade task flow:

1. On the 2018 deployment:

- a. Backup each 2018 subsystem.
 - b. For the management subsystem, use the provided script to extract the 2018 database.
 - c. Power off the 2018 cluster.
2. On a new v10 installation:
 - a. Create a new v10 cluster using the 2018 project directory, the v10 installation utility (**apicup**) and the v10 distribution images (.ova).
 - b. Install the management subsystem. Start the v10 system using the newly created ISO.
 - c. Use a provided script to load the extracted v2018 database info into the new v10 database.
 - d. Install the Developer Portal subsystem and run a restore to load the 2018 backup data.
 - e. Install the Analytics subsystem and run a restore to load the 2018 backup data.
 - f. Upgrade the DataPower Gateway.

Planning a v10 deployment

Before you move from 2018 to v10, plan your v10 deployment by reviewing requirements and features:

- [IBM API Connect Version 10 software product compatibility requirements](#)
- [Requirements for initial deployment on VMware](#)
- [Planning your deployment](#)
- [What's New in the latest release](#) and [What's new for v10 installation](#).
- [Version 10 known limitations](#)

Required backups of 2018 subsystems

Subsystems on 2018 must be backed up in the following order:

1. Management subsystem
2. Developer Portal subsystem
3. Analytics subsystem

Ensure that your archives are valid.

Note: You cannot use a VMware snapshot to skip the preparation steps for the upgrade. VMware snapshots do not provide the necessary API Connect data and backups in a form that can be applied to a new API Connect v10 deployment in a VM.

Required disk space for extraction of 2018 management subsystem data

The upgrade from 2018 to v10 includes extraction of management database data. Prior to running the extract job, ensure that your virtual machine has sufficient disk space to hold a file containing this data.

The amount of disk space required scales with the amount of data on your system. As a minimum, ensure you have 3 times the size of the backup file for your 2018 Cassandra database.

To locate the backup file for your 2018 Cassandra database, see Step 3 in [Restoring the management database in a Kubernetes environment](#).

Automatic shortening of subsystem names with greater than 63 characters

Version 10 imposes a length restriction of 63 characters for subsystem names. Version 2018 did not impose the restriction. If your v2018 subsystem names are greater than 63 characters in length, you must shorten them prior to upgrading. If you do not shorten them, the upgrade will programmatically shorten them for you. You will then want to check `apicconnect-up-v10.yaml` to determine the new shortened name.

The new value is found in `spec.subsystems[n].metadata.name`. For example, for a management subsystem, see the `name: abcde-management` entry below:

```
apiVersion: v1
kind: apic.ibm.com/APIConnectCluster
metadata:
  creationTimestamp: "2021-03-04T13:52:50Z"
  name: abcde
spec:
  secret-name: abcde-my-secret
  subsystems:
  - apiVersion: v1
    kind: apic.ibm.com/ManagementSubsystem
    metadata:
      creationTimestamp: "2021-03-04T13:52:50Z"
      name: abcde-management
    spec:
      ApplianceProperties:
        data-device: sdb
```

Upgrade Limitations

The upgrade is designed to bring your 2018 configuration and data onto a new v10 deployment. Note the following limitations:

- The location of the v10 backups for the management subsystem must be different from the location of the 2018 backups of the management subsystem. Note that for v10, the management database changes from Cassandra to Postgres, so a new backup configuration is required.
- The upgrade does not capture customizations to your 2018 configuration that you made with an extra-values file. If you created an extra-values file on 2018, you must modify it for the v10 deployment, as follows:
 - For the management subsystem, the upgrade process uses a temporary extra-values file to configure upgrade settings, and then removes the temporary file upon successful completion of the upgrade. You can then recreate your extra-values file. See also [Upgrading Management subsystem from 2018 to v10](#)
 - For the Developer Portal subsystem and the Analytics subsystem, if you have an extra-values file on Version 2018, you must convert the file format for Version 10, prior to deploying the Version 10 subsystem. See the instructions in [Upgrading Developer Portal from 2018 to v10](#) and [Upgrading Analytics from 2018 to v10](#)

- Version 10.0.5.0 and later use a new version of the Analytics subsystem, which has the following impacts on your upgrade process:
 - If you want to preserve your v2018 analytics data, export it to a backup location before upgrading as explained in [Preparing the 2018 deployment for upgrade](#).
 - Disable analytics before upgrading as explained in [Preparing the 2018 deployment for upgrade](#).
 - During the upgrade, the v2018 Analytics subsystem cannot work with v10 Management subsystem, so you must upgrade the new Analytics subsystem immediately. If there is an upgrade failure with Analytics, resolve it as soon as possible and run the upgrade again.

x-ibm-gateway-cors-allow-credentials-when-no-cors-policy API property is deprecated

In 2018, when cross-origin resource sharing (CORS) is enabled for an API, every response to a CORS request contains the header:

```
Access-Control-Allow-Credentials: true
```

Inclusion of Access-Control-Allow-Credentials (ACAC) does not provide optimal security, so starting with 2018.4.1.17, an API property can be set to prevent this header from being returned:

```
Name: x-ibm-gateway-cors-allow-credentials-when-no-cors-policy
Collection: *
Value (string): off
```

The `Collection` setting refers to a specific Catalog to which a property value applies. The "*" value means the value applies to all catalogs. You can either specify * or a Catalog (collection) name.

The property can also be set by using the DataPower CLI:

```
api-definition myAPI
  property x-ibm-gateway-cors-allow-credentials-when-no-cors-policy * off
exit
```

Important: The value of this property must be exactly the string `off` (case-sensitive) in order to remove the header.

It is only possible to turn off this header, or leave it on, for an entire API. In order to have more control, such as returning the header for some origins and not others, you must upgrade to v10 and use the CORS policy feature. For more info, see:

- [Enabling CORS support for an API](#)
- OpenAPI 3.0 editor: [Setting API properties](#)
- OpenAPI 2.0 editor: [Setting API properties](#)

In API Connect 10.0.1.4-ifix1-eus or later, the `x-ibm-gateway-cors-allow-credentials-when-no-cors-policy` property works, but only for API definitions which do NOT contain a cors-policy. You cannot define both this property and a CORS Policy on the same API definition.

This property is deprecated in v10: a warning will be written to the log for every CORS request received by an API whose definition contains this property.

For version 10, the recommended way to ensure the ACAC header does not appear in CORS responses is to add a CORS policy to the API definition. In version 10, any API definition which has a cors-policy defined will NOT return the ACAC header by default. Note that it is possible in the CORS policy rules to explicitly return the ACAC header for some or all origins.

If you move your configuration from 2018 to 10.0.1.4-ifix1-eus or later, and have this custom property defined in your API definitions, you will not see any change in behavior, other than new warnings in the log. This property will remove the ACAC header for any API definition where it appears.

If, when you move your configuration from 2018 to 10, you want to have the ACAC header present in the CORS responses, you can leave the API definitions as-is (without the new property defined) and the ACAC header will be returned, as long as a cors-policy is not added to the API definition. Note also that even with a CORS policy defined for an API, the ACAC header can be returned if desired by explicitly allowing it in the rules of the CORS policy.

Note that the `x-ibm-gateway-cors-allow-credentials-when-no-cors-policy` property in an API definition:

- Will cause a commit-time error in version 10 if a cors-policy is also present in the same API definition
- Will NOT work in 10 or 2018 to turn ON the ACAC header; it can ONLY be used to turn OFF the header. Explanation:
 - In 2018, ACAC is on by default
 - In 10 without cors-policy, ACAC is on by default
 - In 10 with cors-policy, ACAC can be turned on for specific origins via rules in cors-policy

Getting started with the upgrade procedure

The first task in the upgrade procedure is to prepare your 2018 deployment for upgrade. To get started, go to [Preparing the 2018 deployment for upgrade](#)

Preparing the 2018 deployment for upgrade

Prepare your 2018 deployment for upgrade to the latest v10 on VMware.

Before you begin

Before you move from 2018 to v10, ensure that you have reviewed [Upgrade from v2018: overview and considerations](#)

Attention: Review the IBM Support page [API Connect: v2018 to v10 upgrade paths](#) to determine whether your Version 2018 deployment can be directly upgraded to the latest release of API Connect.

About this task

On your v2018 deployment, gather the API Connect information that is needed to restore the 2018 configuration onto a new v10 deployment.

Procedure

1. Ensure that the management subsystem is on a 2018 version that is supported for upgrading to the target version, as listed in [Upgrade from v2018: overview and considerations](#).

To view the version, ssh to the VM and run `sudo -i`. Use `apic version`. For example:

```
# apic version
INFO[0000] Log level: info
.
.
.
Packages:
.
.
.
- subsystem-management:<version_number>
```

2. Make sure the API Connect cluster is healthy. For each subsystem (Management, Developer Portal, and Analytics) run the following command:

```
apic health-check
```

If the cluster is healthy, the health-check command returns silently with a 0 return code.

3. Remove any stale upgrade files:

```
apic clean-upgrade-files
```

4. Complete a full backup for the API Connect v2018 Management, Portal, and Analytics subsystems. See [Backing up and restoring](#).

Tip: Ensure your backup archives are valid. For example:

```
$ tar -tzf <FILENAME>.tar.gz >/dev/null
```

On success, the command returns to the command line prompt with no output. If there is a problem with the file, an error similar to the following would be reported:

```
tar: 0511-169 A directory checksum error on media; -559038737 not equal to 67008.
```

5. On the v2018 management subsystem deployment, use the `apicops` utility to ensure the v2018 management database is consistent: Note that `apicops` is distinct from `apicup`.

Important: Upgrade will fail if the `apicops` command is not successfully completed within 2 hours of starting the management upgrade. If the management upgrade must be run more than once, you must run the command in fix mode each time within 2 hours of starting the management upgrade.

- a. Obtain the latest v2018 release of `apicops` from <https://github.com/ibm-apiconnect/apicops#latest-v2018-release>.
- b. Download the `apicops` binary to one of the Management cluster servers, and make it executable:

```
chmod +x apicops
```

See also `apicops` set up requirements: <https://github.com/ibm-apiconnect/apicops#requirements>

- c. Run `apicops` as root:

```
# sudo -i
# apicops preupgrade -m fix --upgrade_to v10 -l <outputLogFileName>
```

- The `-m` option runs the utility in fix mode. Running the utility in fix mode resolves most problems. If the utility identifies a problem it cannot resolve, contact IBM Support.
- Set the `-l` option to send output to a log file.

- d. Confirm from the logs that all issues are fixed.

6. Take a backup of the Management subsystem. This backup is in addition to the backup taken in 4. The `apicops preupgrade` operation modifies the Management database and so it is recommended to have backups from before and after.

7. Make sure cluster is healthy before proceeding:

```
apic health-check
```

If the cluster is healthy, the health-check command returns silently with a 0 return code.

8. Obtain the needed upgrade script and supporting files.

- a. Go to the [What's New in the latest version](#) information page.
- b. Locate the Note: *You can access the latest files from <URL link>*. Select the *<URL link>* to go directly to the Announce page on Fix Central, where you can download files for the latest version of API Connect.
- c. Download **IBM® API Connect <version> scripts and assets required for v2018 > <v10.0.1.x_version> side-by-side upgrades** from API Connect Fix Central.
- d. Decompress the tar file and copy (`scp`) the `dr-upgrade` directory to the v2018 management system. The directory contains `dr-upgrade-extract.py`, which you will use to extract v2018 configuration.

9. Ensure that traffic to the management endpoints is stopped before running the python script.

For example, a load balancer might need to mark the backends as down so that no incoming traffic is accepted.

10. Disassociate and delete your v2018 Analytics services.

Note: Upgrading to v10 will result in the deletion of existing analytics data. For instructions on exporting data from v2018, see the v2018 documentation: <https://www.ibm.com/docs/en/api-connect/2018.x?topic=analytics-exporting-api-event-data>. Also, you can set up analytics offload so that all new analytics data is sent to a third-party system for storage: <https://www.ibm.com/docs/en/api-connect/2018.x?topic=cloud-configuring-analytics-offload-api-connect>.

- a. In Cloud Manager UI, click Topology.
- b. In the section for the Availability Zone that contains the Analytics service, locate the Gateway service that the Analytics service is associated with.
- c. Click the actions menu, and select Unassociate analytics service.

- Remember to disassociate each Analytics service from all Gateways.
- d. In the section for the Availability Zone that contains the Analytics services, locate each Analytics service and click Delete.
11. Extract the v2018 data that is needed for upgrade. Change directory to the **dr-upgrade** directory and run the python extract script as **root**:

```
# sudo -i
# python3 dr-upgrade-extract.py
```

Tip: You can optionally run the script in the background:

```
nohup python3 dr-upgrade-extract.py &> extract.log &
tail -f extract.log
```

- The script creates **2018-extracted-data.zip**. Example output:

```
.
.
.
Extract job is in Running state.
Extract job is complete
Time elapsed: 0:06:17.397687
Zipped 2018 extracted data and logs under '2018-extracted-data.zip'
```

The amount of time required for the extraction is a function of the amount of data in the v2018 database.

The file **2018-extracted-data.zip** contains your application data in an unencrypted format. This data should be kept secure. The data is required to successfully upgrade to v10.

- If the extract script encounters an error:
 - Run the cleanup script:

```
python3 dr-upgrade-cleanup.py
```

- Resolve the errors.
- Re-run the extract script:

```
python3 dr-upgrade-extract.py
```

12. Save the following files to a location where you can access them later, while installing the v10 system:
 - **2018-extracted-data.zip**
 - The **apicup** project directory. You must use this project directory on your v10 system.
13. Power off the virtual machine with the 2018 system.
14. Continue with [Upgrading v2018 subsystems to v10](#).

Upgrading v2018 subsystems to v10

Install the new v10 system and configure each subsystem with configuration information from 2018.

Before you begin

You must have completed [Preparing the 2018 deployment for upgrade](#).

About this task

To upgrade the Management subsystem from 2018 to the latest v10 version, you will do a fresh install of v10, and then run a script to load management data from 2018. For the Developer Portal subsystem and the Analytics subsystem, you will do a fresh install of each subsystem; for Portal only, you will then restore data from the 2018 backup.

When all subsystem upgrades are complete, you can upgrade DataPower Gateway. Optionally, you can then install the API Connect toolkit and the Lightweight Test Environment.

Procedure

1. Obtain the API Connect files from IBM Fix Central.

Note: When upgrading from 2018 to the latest v10 version, you will use the same Fix Central files as are used for a **new** API Connect installation. Do not use the separate product distribution files for **upgrade** of each subsystem on VMware.

The following files are used for upgrading from 2018 on VMware:

 - IBM® API Connect <version> Management for VMware Management subsystem files
 - IBM® API Connect <version> Developer Portal for VMware Developer Portal subsystem files
 - IBM® API Connect <version> Analytics for VMware Analytics subsystem files
 - IBM® API Connect <version> Install Assist for <operating_system_type> The apicup installation utility. Required for all installations on VMware.
2. Install the installation utility.
 - a. Locate the apicup installation utility file for your operating system.
 - b. Rename the file for your OS type to **apicup**. Note that the instructions in this documentation refer to **apicup**.
 - c. Make **apicup** accessible on your operating system:

Operating system	Configuration
------------------	---------------

Operating system	Configuration
OSX and Linux®	i. Make the apicup file an executable file: <code>chmod +x apicup</code> ii. Set your path to the location of your apicup file: <code>export PATH=\$PATH:/Users/your_path/</code>
Windows:	Set your path to the location of your apicup file: <code>set PATH=c:\your_path;%PATH%</code>

- Use the following links to upgrade the subsystems, in the specified order:
 - [Upgrading Management subsystem from 2018 to v10](#)
 - [Upgrading Developer Portal from 2018 to v10](#)
 - [Upgrading Analytics from 2018 to v10](#)
- Upgrade the DataPower Gateway. See [Upgrading DataPower Gateway Service](#). DataPower Gateways requires a firmware upgrade from 2018 to 10 after the upgrade of the other API Connect subsystems is complete.
- Optionally, install new versions of other API Connect packages:
For each of the following distribution packages there is no upgrade process. You replace the existing tools or binaries with the new versions for the latest version of v10.

IBM® API Connect <version> Toolkit for <operating_system_type>
Toolkit command line utility. Packaged standalone, or with API Designer or Loopback:

 - IBM® API Connect <version> Toolkit for <operating_system_type>
 - IBM® API Connect <version> Toolkit with Loopback for <operating_system_type>
 - IBM® API Connect <version> Toolkit Designer with Loopback for <operating_system_type>

Not required during initial installation. After installation, you can download directly from the Cloud Manager UI and API Manager UI. See [Installing the toolkit](#).

IBM® API Connect <version> Local Test Environment
Optional test environment. See [Testing an API with the Local Test Environment](#)

IBM® API Connect <version> Security Signature Bundle File
Checksum files that you can use to verify the integrity of your downloads.
- When you complete the installation and configuration, best practice is to immediately take backups of the new v10 deployment. See:
 - [Backing up the management subsystem](#)
 - [Backing up the Developer Portal subsystem](#)
 - [Running a backup of the Analytics database](#)
 - [Upgrading Management subsystem from 2018 to v10](#)
Upgrade the management subsystem to v10 by installing the latest v10 management subsystem and loading the management database information from 2018.
 - [Upgrading Developer Portal from 2018 to v10](#)
Upgrade the Developer Portal by installing the latest v10 with the 2018 configuration and restoring the subsystem from backups.
 - [Upgrading Analytics from 2018 to v10](#)
Upgrade the Analytics subsystem by installing the v10 subsystem.

Upgrading Management subsystem from 2018 to v10

Upgrade the management subsystem to v10 by installing the latest v10 management subsystem and loading the management database information from 2018.

Before you begin

You should have previously completed the following steps in [Upgrading v2018 subsystems to v10](#):

- Download **IBM® API Connect <version> scripts and assets required for v2018 > <Latest_v10_version> side-by-side upgrades** and **IBM® API Connect <version> Management for VMware** from Fix Central.
- Install **apicup** installation utility.

About this task

Install the latest v10 management subsystem, load the 2018 management database information, and run a cleanup script after the load completes.

Procedure

- Use the v10 version of **apicup** to map your 2018 configuration settings from `apiconnect-up.yml` to the v10 configuration file `apiconnect-up-v10.yaml`:

```
apicup subsys get <subsystem_name>
```

Note: If your subsystem name on 2018 was greater than 63 characters, and you did not manually shorten it prior to upgrading, the upgrade shortened it automatically. To view the new subsystem name:

```
apicup subsys list
```

For more information, see [Automatic shortening of subsystem names with greater than 63 characters](#)

- Review the output from the command. Note the following:
 - Some settings will be automatically converted from their 2018 values, but you should review all values to ensure they are still valid.
- Update the license value by running the following command:

```
apicup licenses accept <License_ID>
```

The <License_ID> is specific to the API Connect program name that you purchased. To view the supported license IDs, see [API Connect licenses](#).

c. Use **apicup** to provide a completely new backup location for the management subsystem in v10. See [Reconfiguring backup settings for the management subsystem](#).

Important:

- For the management subsystem, you cannot use the same backup locations as were used for 2018.
 - For s3 backups you must specify a different s3 bucket for the version 10 deployment, to avoid having version 2018 backups mixed with version 10 backups for the management server.
 - For SFTP backups, create a different folder for version 10 backups, or move the 2018 backups to another location.
- For v10, there is a new (additional) setting **database-backup-s3provider**. This settings is used with S3 backups. This value must be set before install, if S3 backups are being used. The management subsystem might not come online without this parameter being set.

2. Set a temporary configuration to support loading the v2018 data:

a. Place the following text in to a new yaml file, with a name of your choosing:

```
spec:
  template:
  - name: apim
    enabled: false
  - name: apim-schema
    enabled: false
  - name: apim-data
    enabled: false
  - name: taskmanager
    enabled: false
  - name: lur
    enabled: false
  - name: lur-schema
    enabled: false
  - name: lur-data
    enabled: false
  - name: analytics-proxy
    enabled: false
  - name: portal-proxy
    enabled: false
  - name: billing
    enabled: false
  - name: juhu
    enabled: false
  - name: websocket-proxy
    enabled: false
  - name: ui
    enabled: false
```

b. Set the new yaml file as an extra-values configuration file:

```
apicup subsys set <management_subsystem_name> extra-values-file=<path-to-extra-values-file>/<your_extra_values_file>.yaml
```

3. Create your ISO file

```
apicup subsys install mgmt --out mgmtplan-out
```

The **--out** parameter and value are required.

In this example, the ISO file is created in the `myProject/mgmtplan-out` directory.

4. Deploy the ISO and **IBM® API Connect <version> Management for VMware** ova file. Go to [Deploying the Management subsystem OVA file](#).

Note: Wait for the cluster to form - this can take some time.

5. Verify the installation:

```
apicup subsys health-check <management_subsystem_name>
```

Note: This is not a standard install. Some services are disabled during this process, so there will be some pods missing. As a result, you cannot use the Cloud Manager UI to verify the management subsystem until the data from v2018 is loaded.

6. Copy (**scp**) the entire **dr-upgrade** .tgz to the v10 management system VM. Uncompress the file (**tar -xzf**). Verify that the directory contains these files:

- **dr-upgrade-load.py**
A script to load the v2018 data.
- **dr-upgrade-cleanup.py**
A script to cleanup after load of the v2018 data
- **2018-extracted-data.zip**
The data from your v2018 deployment

The directory contains other template files that are used by the scripts.

7. Now load 2018 data into v10 database.

a. ssh into the appliance as **root**.

b. Change directory to the **dr-upgrade** directory

The **dr-upgrade-load.py** script requires **2018-extracted-data.zip** to be in the same directory

c. Run the python script **dr-upgrade-load.py**:

```
# sudo -i
# python3 dr-upgrade-load.py
```

Tip: You can optionally run the script in the background:

```
nohup python3 dr-upgrade-load.py &> load.log &
tail -f load.log
```

- The length of time needed for the load script to complete scales with the size of the data in your system. The upgrade-load job creates a Kubernetes pod named `apiconnect-v10-upgrade-load-<xxxxxx>`. You can follow the Kubernetes logs if you want to track the progress of the script.
- The python script outputs the location (`volume_path`) of the data/logs directories to backup.
- Once `dr-upgrade-load.py` returns successfully the load is complete.

Note: If `dr-upgrade-load.py` encounters an error:

- `ssh` into the appliance as root, and gather the upgrade logs. Check pod logs and PV logs (`volume_path`). To create a zip file with the contents of directories with the logs:

```
zip -r upgrade-pvc.zip {{volume_path}}
```

- Contact IBM Support for guidance on how to resolve the errors.
- After the errors are resolved, run the clean up script:

```
python3 dr-upgrade-cleanup.py --load-error
```

Be sure to include the required parameter `--load-error`. Note that the health-check will fail until the load script is run again in the next step.

- Re-run `dr-upgrade-load.py`

```
python3 dr-upgrade-load.py --rerun
```

Be sure to include the required parameter `--rerun`.

8. Wait for the completion of the job and then return to your local machine (used for configuration) and use `apicup` to unset the extra-values file To unset:

```
apicup subsys set <management_subsystem_name> extra-values-file=
```

9. Set an extra-values file if needed for your customizations:

```
apicup subsys set <management_subsystem_name> extra-values-file=<your_extra-values-file>
```

For example, see [Setting rate limits for public APIs on the management service](#).

10. Install the subsystem:

```
apicup subsys install <management_subsystem_name>
```

11. Verify all management resources have come up

```
apicup subsys health-check <management_subsystem_name>
```

If the subsystem is healthy, the health-check command returns silently with a 0 return code.

If an error is encountered, see [Upgrade Limitations](#).

Your load of v2018 data is now complete.

12. When the `dr-upgrade-load` job has succeeded, backup the PVC data and logs from the `dr-upgrade-load` job:

- a. `ssh` into the appliance as root.
- b. Find the path to the volume:

```
kubectl get pv | grep 'pv-claim-apiconnect-v10-upgrade' | awk '{print $1}' | xargs kubectl get pv -ojsonpath='{.spec.local.path}'
```

- c. Zip up the directory:

```
zip -r upgrade-pvc.zip {{volume_path}}
```

- d. Run the cleanup script:

```
python3 dr-upgrade-cleanup.py
```

This script cleans up the following:

- Load job `apiconnect-v10-upgrade-load`
- Load PVC `pv-claim-apiconnect-v10-upgrade`
- `2018-extracted-data.zip`
- Unpacked `2018-extracted-data.zip` from the PV

13. Restart all `nats-server` pods.

- a. SSH into the server:

- i. Run the following command to connect as the API Connect administrator, replacing `ip_address` with the appropriate IP address:

```
ssh ip_address -l apicadm
```

- ii. When prompted, select Yes to continue connecting.

- iii. When you are connected, run the following command to receive the necessary permissions for working directly on the appliance:

```
sudo -i
```

- b. Restart all `nats-server` pods by running the following command:

```
kubectl -n <namespace> delete po -l app.kubernetes.io/name=natscluster
```

14. Next, upgrade the other subsystems. Return to [Upgrading v2018 subsystems to v10](#).

Upgrading Developer Portal from 2018 to v10

Upgrade the Developer Portal by installing the latest v10 with the 2018 configuration and restoring the subsystem from backups.

About this task

You will verify your configuration, create an ISO, deploy the v10 OVA file into the VM, and restore your Developer Portal data from backup.

You do not have to specify configuration settings because the Portal configuration is defined in the 2018 project directory, which you will continue to use for v10. However, you must review the backups settings.

Procedure

1. Obtain **IBM® API Connect <version> Developer Portal for VMware** from Fix Central, as described in [Upgrading v2018 subsystems to v10](#).
2. Use the version 10 release of `apicup` to map your 2018 configuration settings from `apiconnect-up.yml` to the v10 configuration file `apiconnect-up-v10.yml`:

```
apicup subsys get <subsystem_name>
```

- a. Review the output from the command, and ensure backup settings from 2018 have been set or migrated to the configuration for version 10. See [Configuring backup settings for the Developer Portal subsystem](#).
- b. If your 2018 deployment included an extra-values-file, you must first convert the format of the file for v10. For example, if you had an extra-values-file to support multiple Portal endpoints, review the new format in the version 10 documentation. See [Defining multiple portal endpoints for a VMware environment](#).

Note: If your subsystem name on 2018 was greater than 63 characters, and you did not manually shorten it prior to upgrading, the upgrade shortened it automatically. To view the new subsystem name:

```
apicup subsys list
```

For more information, see [Automatic shortening of subsystem names with greater than 63 characters](#)

- c. Update the license value by running the following command:

```
apicup licenses accept <License_ID>
```

The `<License_ID>` is specific to the API Connect program name that you purchased. To view the supported license IDs, see [API Connect licenses](#).

3. Create your ISO file.

```
apicup subsys install port --out portplan-out
```

The `--out` parameter and value are required. In this example, the ISO file is created in the `myProject/portplan-out` directory.

4. Deploy the file into the VM. See [Deploying the Developer Portal subsystem OVA file](#).
5. Verify the deployment: See [Verifying deployment of the Developer Portal subsystem](#).
6. Restore Portal using the backup taken on 2018. See [Restoring the Developer Portal subsystem](#).
7. Confirm the restore has completed successfully:

```
apicup subsys health-check <portal_subsystem_name>
```

When no problems are detected, the health-check commands returns silently with a 0 return code.

Important: If the health check command states that a reboot is required, you must ensure that all Portal sites have been upgraded first before performing the reboot. To check that the Portal sites have upgraded, inspect the `platform` and `state` values for each site. The sites should all be in **INSTALLED** state, and have `platform` equal to the new platform that you can identify with `platforms:list`. Note that there will be two platforms present in the list while the site upgrades are happening. There should be only one `platform` after they have completed. To check that the Portal sites were upgraded:

- a. Obtain the portal service id and endpoint:

```
apic portal-services:get -o admin -s <management_server_endpoint> \
  --availability-zone availability-zone-default <portal-service-name> \
  --output - --format json
```

- b. List the sites:

```
apic --mode portaladmin sites:list -s <management_server_endpoint> \
  --portal_service_name <portal_service_name_from_above_command> \
  --format json
```

Any sites currently upgrading will be listed as **UPGRADING**. Once all sites have finished upgrading they should have the **INSTALLED** status and the new platform version listed.

See also: [apic sites:list](#) and [Using the sites commands](#).

- c. After all sites are in **INSTALLED** state and have the new platform listed, run:

```
apic --mode portaladmin platforms:list -s <management_server_endpoint> \
  --portal_service_name <portal_service_name_from_above_command> \
  --format json
```

The new version of the platform should be the only platform listed.

See also: [apic platforms:list](#) and [Using the platforms commands](#).

If an error is encountered, see [Upgrade Limitations](#).

8. Return to [Upgrading v2018 subsystems to v10](#).

Upgrading Analytics from 2018 to v10

Upgrade the Analytics subsystem by installing the v10 subsystem.

About this task

You will verify your configuration, create an ISO, and deploy the v10 OVA file into the VM. Due to changes in the v10 Analytics subsystem, you cannot restore your Analytics data from backup. You can configure Analytics settings after the upgrade is complete.

Procedure

1. Obtain **IBM® API Connect <version> Analytics for VMware** from Fix Central, as described in [Upgrading v2018 subsystems to v10](#).
2. Use the v10 version of **apicup** to map your 2018 configuration settings from **apiconnect-up.yml** to the v10 configuration file **apiconnect-up-v10.yaml**:

```
apicup subsys get <subsystem_name>
```

Note: If your subsystem name on 2018 was greater than 63 characters, and you did not manually shorten it prior to upgrading, the upgrade shortened it automatically. To view the new subsystem name:

```
apicup subsys list
```

For more information, see [Automatic shortening of subsystem names with greater than 63 characters](#)

Complete the following steps:

- a. Review the output from the command.
- b. Specify backup settings for v10. You cannot use the settings from 2018. See [Configuring backup settings for Analytics](#).
- c. If your 2018 deployment included an extra-values-file, you must first convert the format of the file for v10. To review creation of extra-values files for v10, see [Creating the analytics-extra-values file](#).
- d. Update the license value by running the following command:

```
apicup licenses accept <License_ID>
```

The <License_ID> is specific to the API Connect program name that you purchased. To view the supported license IDs, see [API Connect licenses](#).

3. Create your ISO file.

```
apicup subsys install analyt --out analytplan-out
```

The **--out** parameter and value are required. In this example, the ISO file is created in the myProject/analytplan-out/node-config directory.

4. Deploy the file into the VM. See [Deploying the Analytics subsystem OVA file](#)
5. Verify the deployment. See [Verifying deployment of the Analytics subsystem](#)
6. Confirm the restore has completed successfully:

```
apicup subsys health-check <analytics_subsystem_name>
```

When no problems are detected, the health-check commands returns silently with a 0 return code.

If an error is encountered, see [Upgrade Limitations](#).

7. Configure the Analytics settings as explained in [Enabling additional analytics features](#).
8. Return to [Upgrading v2018 subsystems to v10](#).

Enabling Analytics after upgrading

After the upgrade successfully completes, re-enable Analytics data collection.

Before you begin

Skip this task if you are upgrading from v10.0.5.1 or later, or migrating from v2018 to v10.

Procedure

1. Delete your pre-10.0.5.x analytics storage allocation. If you skip this step then your previous analytics data, although not extractable, continues to occupy disk space.
 - a. SSH to each of your Analytics VMs as the user 'apicadm', for example: **ssh apicadm@analytics1.example.com**.
 - b. Sudo to the root user: **sudo -i**.
 - c. Identify the pre-10.0.5.x analytics storage allocation (known as PVCs) by running:



```
kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
data-ger2-analytics-storage-0	Bound	local-pv-8735e017	250Gi	RWO	local-storage	3h29m
data-ger2-analytics-storage-1	Bound	local-pv-2d14de06	250Gi	RWO	local-storage	3h29m
data-ger2-analytics-storage-2	Bound	local-pv-b2a8a5d	250Gi	RWO	local-storage	3h29m
data-ger2-analytics-storage-shared-0	Bound	local-pv-30aac7ed	250Gi	RWO	local-storage	227d
data-ger2-analytics-storage-shared-1	Bound	local-pv-e4a82f28	250Gi	RWO	local-storage	227d
data-ger2-analytics-storage-shared-2	Bound	local-pv-82018f91	250Gi	RWO	local-storage	227d

- d. Delete all PVCs with 'storage' in their name that date back to before you started your upgrade. In this example these are the PVCs that are 227 days old.

```
kubectl delete pvc <pvc name>
```

2. Add the new analytics:view permission to the Administrator role in the Cloud Manager UI.

- a. Click , then select Roles
- b. Click  on the Administrator role and select Edit
- c. Select the checkbox next to the analytics permission and click save:

Analytics


View

3. Register new analytics services: [Registering an analytics service](#)

4. Associate your new analytics services with your gateways [Associating an analytics service with a gateway service](#)

Troubleshooting upgrades on VMware

Review the following troubleshooting guidance if you encounter a problem while upgrading API Connect on VMware.

Note: In the  Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.

- [Appliance upgrade from 10.0.4.x to 10.0.5.3 and 10.0.5.4 fails](#)
- [The original primary PVC is not attached to any pods in this namespace](#)
- [Original PostgreSQL primary not found](#)
- [The cloud-final.service fails during upgrade](#)
- [Upgrade to API Connect 10.0.5.3 fails to start because management subsystem fails the health check](#)
- [Postgres replica fails to start due to permissions issue](#)
- [Upgrade stuck with "Unable to upgrade appliance-base: exit status 100" message](#)
- [Some containers in the kube-system namespace show a status of ErrImageNeverPull](#)
- [Pod stuck in Pending status during upgrade](#)
- [Database replica pods stuck in Unknown or Pending state](#)
- [etcd pod stuck in Terminating state](#)
- [Management subsystem upgrade imagePullBackOff issue](#)
- [Postgres pods fail to start after upgrade](#)
- [Upgrading a 3 node profile to IBM API Connect 10.0.3.0 or later might result in some portal-db/www.pods being stuck in the Pending state](#)
- [Issues when installing Drupal 8 based custom modules or sub-themes into the Drupal 9 based Developer Portal](#)

Appliance upgrade from 10.0.4.x to 10.0.5.3 and 10.0.5.4 fails

A direct upgrade from 10.0.4.x to 10.0.5.3 or to 10.0.5.4 on an appliance might fail with a message similar to the following example:

```
Error: failed to install the subsystem: unable to close stream to remote server: rpc error: code = DeadlineExceeded desc = context deadline exceeded
```

Correct the problem by completing the following steps:

1. Restore the content of the `/usr/lib/python3/dist-packages/cloudinit/config/cc_apiconnect.py` file to the original 10.0.4.x definition as shown in the following example:

```
from cloudinit.settings import PER_ALWAYS
from yaml import dump
from subprocess import call
import time

frequency = PER_ALWAYS

UNLOCK_ATTEMPT_MAX_COUNT = 50
UNLOCK_ATTEMPT_DELAY = 10

def handle(name, _cfg, _cloud, log, _args):
    if (name == 'apiconnect'):
        f = open('/var/lib/cloud/instance/apiconnect.yml', 'w')
        f.write(dump(_cfg["apiconnect"]))
        f.close()

        for i in range(1, UNLOCK_ATTEMPT_MAX_COUNT):
            rc = call(["/usr/bin/apic", "unlock", "--prompt=false"])
            if rc == 0:
                return
            log.warning("apic unlock attempt %s of %s failed", i, UNLOCK_ATTEMPT_MAX_COUNT)
            time.sleep(UNLOCK_ATTEMPT_DELAY)
        log.warning("Use apic unlock to mount the secure partition and start the node")
```

2. Reboot the appliance.

The original primary PVC is not attached to any pods in this namespace

If the `apicops version:pre-upgrade` check reports

```
The original primary PVC is not attached to any pods in this namespace
```

and you have not started the upgrade yet (you have not run the **apicup subsys install** command) then follow these steps:

1. Take a backup of your management subsystem: [Backing up and restoring the Management subsystem](#).
2. Restore the new backup: [Restoring the management subsystem](#).

The action of taking and restoring a management backup fixes the problem that causes the error message. If you see this error after starting upgrade, see [Original PostgreSQL primary not found](#).

Original PostgreSQL primary not found

If you ran the **apicup subsys install <tar file>** command to start an upgrade, the upgrade does not complete, and the **apicup health-check** command returns:

```
apicup subsys health-check management
Error: Cluster not in good health:
expect member apicdev.ibm.com 'Install stage' to be 'Done' (actual: SETUP_SUBSYSTEM) | Detail: reconcile.go#apply subsystem not ready
```

then SSH into the management appliance and switch to root user to get more details of the error:

```
ssh apicadm@<management appliance>
sudo -i
```

Run **kubectl describe mgmt**, and check the output for the following message:

```
kubectl describe mgmt
...
Original PostgreSQL primary not found, further operand upgrade will be blocked. Set apiconnect-operator/db-primary-not-found-allow-upgrade=true annotation to unblock operand upgrade and perform db backup and restore after upgrade to restore original primary
...
```

To fix this problem, complete the following steps:

1. Create an extra-values.yaml file and set the annotation:

```
metadata:
  annotations:
    apiconnect-operator/db-primary-not-found-allow-upgrade: "true"
```

Note: If you already have an extra-values.yaml file, update the existing file with above annotation.

2. Set the extra-values-file property on your management subsystem with **apicup**:

```
apicup subsys set <management subsystem name> extra-values-file=<extra values filename>
```

3. Restart the upgrade, with the **--skip-health-check** flag:

```
apicup subsys install <management subsystem name> <upgrade tar file> <control plane file - if required> --skip-health-check
```

4. When upgrade is complete:

- a. Remove the annotation from your extra-values.yaml file.
- b. Take a backup of your management subsystem: [Backing up and restoring the Management subsystem](#).
- c. Restore the new backup: [Restoring the management subsystem](#). The action of taking and restoring a management backup fixes the problem that causes the error message **The original primary PVC is not attached to any pods in this namespace**. Be careful to restore from the backup that is taken after the upgrade, and not from a backup taken before upgrade.

The cloud-final.service fails during upgrade

Sometimes during upgrade, there is an issue with the **cloud-final.service** and a node, and the appliance-manager enters a bad state.

To determine whether your upgrade encountered this problem, complete the following steps:

1. Check the output of **apic health-check** command for a result similar to the following example:

```
# apic health-check
INFO[0000] Log level: info
FATA[0000] Unable to cluster status: rpc error: code = Unavailable desc = connection error: desc = "transport: Error while dialing dial tcp 9.20.153.38:9178: connect: connection refuse"
```

2. Check the response to the **journalctl -u appliance-manager | grep cloud-final** command and see if it looks like the following example:

```
# journalctl -u appliance-manager | grep cloud-final
Nov 21 19:41:58 apimdev1040 apic[2569]: Job for cloud-final.service failed because the control process exited with error code.
Nov 21 19:41:58 apimdev1040 apic[2569]: See "systemctl status cloud-final.service" and "journalctl -xe" for details.
```

If you determined that this was the problem, you can correct it by running each of the following commands on every node:

1. **systemctl restart cloud-final.service**
2. **apic lock**
3. **apic unlock**

Upgrade to API Connect 10.0.5.3 fails to start because management subsystem fails the health check

This issue might be caused by a corrupt compliance entry in the pgbouncer.ini file. When the operator is upgraded to v10.0.5.3 but the ManagementCluster CR is not yet upgraded, the operator might update the pgbouncer.ini file in the PGBouncer secret with the older ManagementCluster CR's profile file, which does not contain any value for the `compliance pool_size`. As a result, the value gets incorrectly set to the string `<no value>`.

When this issue occurs, then the health check does not report the management subsystem as healthy so the upgrade will not start.

The following example shows what the error looks like in the pgbouncer logs:

```
Wed May 17 09:00:02 UTC 2023 INFO: Starting pgBouncer..
2023-05-17 09:00:02.897 UTC [24] ERROR syntax error in connection string
2023-05-17 09:00:02.897 UTC [24] ERROR invalid value "host=mgmt-a516d013-postgres port=5432 dbname=compliance pool_size=<no value>" for parameter compliance in configuration (/pgconf/pgbouncer.ini:7)
2023-05-17 09:00:02.897 UTC [24] FATAL cannot load config file
```

Resolve the issue by completing the following steps:

1. SSH to the root user:

```
ssh apicadm@<vm-hostname>
sudo -i
```

2. Run the commands in the following script:

```
NAMESPACE=<mgmt-namespace>
BOUNCER_SECRET=<management-prefix>-postgres-pgbouncer-secret
TEMP_FILE=/tmp/pgbouncer.ini

**Step 1 - Common: Get the existing pgbouncer.ini file**
kubectl get secret -n $NAMESPACE $BOUNCER_SECRET -o jsonpath='{.data.pgbouncer\.ini}' | base64 -d > $TEMP_FILE

**Step 2 - Linux version: Update the file and use it to patch the Secret on the cluster**
sed 's/<no value>/20/' $TEMP_FILE | base64 -w0 | xargs -I{} kubectl patch secret -n $NAMESPACE $BOUNCER_SECRET --
type='json' -p="{['op' : 'replace' , 'path' : '/data/pgbouncer.ini' , 'value' : {} ]}"

**Step 2 - Mac version: Update the file and use it to patch the Secret on the cluster**
sed 's/<no value>/20/' $TEMP_FILE | base64 -b0 | xargs -S2000 -I{} kubectl patch secret -n $NAMESPACE $BOUNCER_SECRET --
type='json' -p="{['op' : 'replace' , 'path' : '/data/pgbouncer.ini' , 'value' : {} ]}"

# Step 3 - Common: Restart pgbouncer to pick up the updated Secret configuration
kubectl delete pod <bouncer_pod_name> -n $NAMESPACE
```

Postgres replica fails to start due to permissions issue

Symptom:

```
2023-04-11 11:31:40,535 INFO: Lock owner: v10-0-5-up-ce0f24dc-site1-postgres-iszz-c5d9c4f5c-9d9fj; I am v10-0-5-up-ce0f24dc-
site1-postgres-75845474bd-kzcjl
2023-04-11 11:31:40,537 INFO: starting as a secondary
2023-04-11 11:31:41.032 UTC [151610][FATAL: data directory "/pgdata/v10-0-5-up-ce0f24dc-site1-postgres" has invalid
permissions
2023-04-11 11:31:41.032 UTC [151610][DETAIL: Permissions should be u=rwx (0700) or u=rwx,g=rx (0750) .
2023-04-11 11:31:41,055 INFO: postmaster pid=151610
/tmp:5432 - no response
2023-04-11 11:31:41,091 INFO: Lock owner: v10-0-5-up-ce0f24dc-site1-postgres-iszz-c5d9c4f5c-9d9fj; I am v10-0-5-up-ce0f24dc-
site1-postgres-75845474bd-kzcjl
2023-04-11 11:31:41,092 INFO: failed to start postgres
```

Resolve the issue by completing the following steps to correct the permissions and restart the replica pod:

1. Run the following command to exec into the pod:

```
k exec -it <pg-primary-pod> -- bash
```

2. Run the following command to reset the permissions:

```
chmod 0700 /pgdata/<pg-cluster-name>
```

3. Restart the failed pod.

Upgrade stuck with "Unable to upgrade appliance-base: exit status 100" message

This error can occur when the deployment is using an incorrect version of `containerd`. There is a known issue where upgrades from 10.0.1.7-eus or 10.0.1.8-eus to 10.0.5.1 incorrectly upgraded the `containerd` version.

On each node, complete the following steps to determine the version of `containerd` and if necessary, downgrade it to the correct version:

1. Run the following set of commands on the healthy node to downgrade the version of `containerd`:

```
systemctl stop appliance-manager
systemctl stop kubelet
systemctl stop containerd
apt-get --allow-downgrades upgrade -y containerd.io
sed -i 's/KillMode=process/KillMode=mixed/g' /lib/systemd/system/containerd.service
systemctl daemon-reload
systemctl restart containerd
apic lock
apic unlock
```

2. Run the following command to verify that `containerd` now displays `containerd://1.5.11` for the CONTAINER-RUNTIME:

```
kubectl get nodes -o wide
```

Remember to complete these steps on every node.

Some containers in the `kube-system` namespace show a status of `ErrImageNeverPull`

When upgrading, some of the containers in the `kube-system` namespace might show a status of `ErrImageNeverPull`. This happens due to Docker not successfully loading all images from the upgrade `.tgz` file. To resolve this issue and enable the upgrade to proceed, complete the following steps:

1. Run the following command to determine which node is missing the control plane files:

```
kubectl -n kube-system get pods -owide
```

This command returns the names of the nodes containing pods with the `ErrImageNeverPull` status, which indicates missing control plane files.

2. On the node that is missing the control plane files, run the following command to determine which versions of the control plane are missing:

```
cat /var/lib/apiconnect/appliance-control-plane-current
```

3. For each missing control plane, run the following command on the same node to add it, replacing `<version>` with the version of the control plane:

```
docker load < /usr/local/lib/appliance-control-plane/<version>/kubernetes.tgz
```

Pod stuck in `Pending` status during upgrade

When upgrading, the scheduler might deploy a subset of the same microservice pods on the same node. This can prevent other pods with `requiredDuringSchedulingIgnoredDuringExecution` affinity rules from being deployed due to a lack of resources on a subset of nodes. To allow the pending containers to be deployed successfully, identify any pods of the same type that are scheduled on the same node and delete one of them. This will free up space and cause the deleted pod to get rescheduled. To identify pods that are eligible to be deleted, and then delete the pods, complete the following steps:

1. Run the following command and check for any pods of the same type that are on the same node:

```
kubectl get po -o=custom-columns='name:.metadata.name, node:.spec.nodeName, antiAffinity:.spec.affinity.podAntiAffinity.preferredDuringSchedulingIgnoredDuringExecution' | grep -v '<none>' | awk '{print $1" "$2}'
```

In the following example snippet, one of the two `apim` pods on the node `test0186` should be deleted:

```
stv3-management-analytics-proxy-56848c8c69-phpdh test0186
stv3-management-analytics-proxy-56848c8c69-sc45f test0187
stv3-management-analytics-proxy-56848c8c69-scf6g test0188
stv3-management-apim-5574796948-6lnwj test0186
stv3-management-apim-5574796948-h9dgx test0186
stv3-management-apim-5574796948-tsb4g test0188
```

2. Run the following command to delete a pod:

```
kubectl delete po <pod_name>
```

Database replica pods stuck in `Unknown` or `Pending` state

In certain scenarios, a postgres replica pod may not recover to a healthy state when a restore completes, a node outage occurs, or after a fresh install or upgrade. In these cases, a postgres pod remains in a `Unknown` or a `Pending` state after a number of minutes. The pod fail to get into a `Running` state.

This situation occurs when the replicas do not initialize properly. You can use the `patronictl reinit` command to reinitialize the replica. Note that this command syncs the replica's volume data from the current Primary pod.

Use the following steps to get the pod back into a working state:

1. SSH into the VM as root.
2. Exec onto the failing pod:

```
kubectl exec -it <postgres_replica_pod_name> -n <namespace> -- bash
```

3. List the cluster members:

```
patronictl list
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) -----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| fxpk-management-01191b80-postgres-586f899fdf-6s25b | 172.16.172.244 | Leader | start failed | 3 | unknown |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51 | Leader | running | 3 | 0 |
| fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m | 172.16.53.68 | Leader | running | 3 | 0 |
+-----+-----+-----+-----+-----+-----+-----+
```

In the example shown above `fxpk-management-01191b80-postgres-586f899fdf-6s25b` is not in running state.

Note the `clusterName` and `replicaName` which are not up:

- `clusterName` - `fxpk-management-01191b80-postgres`
- `replicaName` - `fxpk-management-01191b80-postgres-586f899fdf-6s25b`

4. Run:

```
patronictl reinit <clusterName> <replicaName-which-is-not-running>
```

Example:

```
patronictl reinit fxpk-management-01191b80-postgres fxpk-management-01191b80-postgres-586f899fdf-6s25b
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) -----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+-----+
```

```

+-----+-----+-----+-----+-----+-----+
| fxpk-management-01191b80-postgres-586f899fdf-6s25b | 172.16.172.244 | | start failed | | unknown | |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51 | Leader | running | 3 | |
| fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m | 172.16.53.68 | | running | 3 | | 0 |
+-----+-----+-----+-----+-----+-----+
Are you sure you want to reinitialize members fxpk-management-01191b80-postgres-586f899fdf-6s25b? [y/N]: y
Success: reinitialize for member fxpk-management-01191b80-postgres-586f899fdf-6s25b

```

5. Run `patronictl list` again.

You may also observe that the replica is on a different Timeline (TL) and possibly have a Lag in MB. It may take a few minutes for the pod to switch onto the same TL as the others and the Lag should slowly go to 0.

For example:

```

bash-4.2$ patronictl list
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) -----+-----+-----+-----+-----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| fxpk-management-01191b80-postgres-586f899fdf-6s25b | 172.16.172.244 | | running | 1 | 23360 | |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51 | Leader | running | 3 | |
| fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m | 172.16.53.68 | | running | 3 | | 0 |
+-----+-----+-----+-----+-----+-----+

```

6. The pod that previously was in an **Unknown** or **Pending** state or **(0/1) Running** state is now in **(1/1) Running** state.

etcd pod stuck in Terminating state

During an upgrade, you might see a health-check reporting the upgrade is stuck in **ETCD** stage and that an **etcd** pod is stuck in the **Terminating** state.

Verify the issue by completing the following steps:

1. SSH into the VM as root.
2. Run the following command and verify that the upgrade stage is **ETCD**:

```
apic status
```

3. Run the following command to determine whether an **etcd** pod is stuck in the **Terminating** state:

```
kubectl get pods -n namespace | grep etcd
```

Resolve the issue by completing the following steps:

1. Run the following command to retrieve the names of the **etcd** pods:

```
kubectl get pods -n namespace -o wide | grep etcd
```

2. On the pod that is stuck, SSH into the VM as root.
3. Run the following command to restart the pod:

```
systemctl restart kubelet
```

Management subsystem upgrade imagePullBackOff issue

When upgrading the management subsystem, for example from v10.0.4.0-ifix3 to v10.0.5.1, the postgres pods might run into an **imagePullBackOff** issue, where the postgres pods are looking for the v10.0.4.0-ifix3 images that exist in the Docker registry, but the images haven't been loaded into the **Containerd** registry. To work around this issue, complete the following steps to manually move each image from the Docker registry to the **Containerd** registry:

1. Save the following Docker images, **ibm-apiconnect-management-k8s-init**, **crunchy-pgbackrest-repo**, **crunchy-pgbackrest**, **crunchy-pgbouncer**, and **crunchy-postgres-ha**, into a .tar file and keep in a secure location. For example, from v10.0.4.0-ifix3:

```
docker save 127.0.0.1:8675/10.0.4.0-ifix3/ibm-apiconnect-management-pgo-scheduler > pgo-scheduler.tar
```

2. Import each saved image into the **Containerd** registry. For example:

```
ctr --namespace k8s.io image import pgo-scheduler.tar --digests=true
```

3. Re-tag the **import-`<datestamp>`@sha256:nnn** images to their appropriate ifix2 tags and digests. For example:

```
ctr --namespace k8s.io image tag <source> <target>
```

where:

- **<source>** is the string that is returned in step 2.
- **<target>** is the missing image; you can determine it by describing the pod that is in the **ImagePullBackFailure** state.

Postgres pods fail to start after upgrade

Note: This topic applies only to upgrading the management subsystem from v10.0.2.0 or later to a newer version of v10.0.x. This problem does not occur when upgrading from v10.0.1.2-ifix2 to v10.0.3.0 or later.

When upgrading the management subsystem from v10.0.2.0 or later, as part of [Upgrading to the latest release on VMware](#), you might encounter an error message when checking the subsystem health upon completion of the upgrade. For example:

```

apic health-check
INFO[0000] Log level: info
FATA[0006] Cluster not in good health:
ManagementCluster (current ha mode: active) is not ready | State: 15/16 Phase: Pending

```

To troubleshoot when a message like this occurs:

1. Check the state of postgres pods:

```
kubectl get pods | grep postgres
```

For example:

```
root@apimdev1146:~# kubectl get pods | grep postgres
fxpk-management-fd8b0b1f-postgres-577594c7f-k54pk          0/1      Init:CrashLoopBackOff   17       22h
fxpk-management-fd8b0b1f-postgres-backrest-shared-repo-7fctp88w  1/1      Running                  2        22h
fxpk-management-fd8b0b1f-postgres-elbx-698f445649-rlc2g     0/1      Init:CrashLoopBackOff   16       22h
fxpk-management-fd8b0b1f-postgres-pgbouncer-64f57b7cc7-52bk8  1/1      Running                  2        22h
fxpk-management-fd8b0b1f-postgres-pgbouncer-64f57b7cc7-jjjvb  1/1      Running                  1        22h
fxpk-management-fd8b0b1f-postgres-pgbouncer-64f57b7cc7-qp4zh  1/1      Running                  2        22h
fxpk-management-fd8b0b1f-postgres-stanza-create-6pt6c       0/1      Completed                0        22h
fxpk-management-fd8b0b1f-postgres-ubba-79ccdd5cc6-kj4zx     0/1      Init:CrashLoopBackOff   17       22h
postgres-operator-85fb96db4b-gk8k8                          4/4      Running                  8        22h
```

2. If any pods show `Init:CrashLoopBackOff` status, restart the pods. To force a restart, delete the pods:

```
kubectl delete pod <name_of_postgres_pod>
```

For example:

```
kubectl delete pod fxpk-management-fd8b0b1f-postgres-577594c7f-k54pk
kubectl delete pod fxpk-management-fd8b0b1f-postgres-elbx-698f445649-rlc2g
kubectl delete pod fxpk-management-fd8b0b1f-postgres-ubba-79ccdd5cc6-kj4zx
```

When pods are deleted, the deployment automatically restarts them.

3. Re-run the health check. For example:

```
apicup subsys health-check <subsys_name>
```

4. When health check is successful, return to the next upgrade step in [Upgrading to the latest release on VMware](#).

Upgrading a 3 node profile to IBM API Connect 10.0.3.0 or later might result in some `portal-db/www` pods being stuck in the `Pending` state

IBM® API Connect 10.0.3.0 introduces the pod anti-affinity required rule, meaning that in a 3 node profile deployment, all 3 db and www pods can run only if there are at least 3 running worker nodes. This rule can cause some upgrades to version 10.0.3.0 or later to become stuck in the `Pending` state, in which case some extra steps are needed during the upgrade to workaround the issue. See the following example for detailed information about the issue, and how to continue with the upgrade.

Important: You must have a backup of your current deployment before starting the upgrade.

To run the commands that are needed to workaround this issue, you must log in to the virtual machine portal subsystem by using an SSH tool.

1. Run the following command to log in as `apicadm`, which is the API Connect ID that has administrator privileges:

```
ssh portal_ip_address -l apicadm
```

Where `portal_ip_address` is the IP address of the portal subsystem.

2. Then get a root shell by running the following command:

```
sudo -i
```

In the following example, the VMware stack has 3 VMs (worker nodes):

```
$ kubectl get nodes
NAME          STATUS    ROLES    AGE   VERSION
apimdev0103   Ready    worker   42m   v1.20.0
apimdev0129   Ready    worker   45m   v1.20.0
apimdev1066   Ready    worker   39m   v1.20.0
```

The pods have been scheduled across only 2 of the 3 worker nodes due to a transient problem with `apimdev1066`, as shown in the following pod list. Pods without persistent storage, such as `nginx-X`, can be rescheduled to `apimdev1066` as soon as they are restarted, but any pods with persistent local storage, such as `db-X` and `www-X`, have to be rescheduled onto the same worker node as that is where their files live.

```
$ kubectl get po -o wide
NAME          READY    STATUS    RESTARTS   AGE   IP             NODE          NOMINATED NODE
READINESS GATES
ejs-portal-nginx-84f57ffd8c-hbf66  1/1      Running   0           5m12s  888.16.109.208  apimdev0103  <none>         <none>
ejs-portal-nginx-84f57ffd8c-mvq96  1/1      Running   0           5m12s  888.16.142.215  apimdev0129  <none>         <none>
ejs-portal-nginx-84f57ffd8c-vpmtl  1/1      Running   0           5m12s  888.16.142.214  apimdev0129  <none>         <none>
ejs-portal-sitel-db-0              2/2      Running   0           4m39s  888.16.109.209  apimdev0103  <none>         <none>
ejs-portal-sitel-db-1              2/2      Running   0           6m37s  888.16.109.206  apimdev0103  <none>         <none>
ejs-portal-sitel-db-2              2/2      Running   0           4m39s  888.16.142.216  apimdev0129  <none>         <none>
ejs-portal-sitel-www-0             2/2      Running   0           4m9s   888.16.109.210  apimdev0103  <none>         <none>
ejs-portal-sitel-www-1             2/2      Running   0           6m37s  888.16.142.213  apimdev0129  <none>         <none>
ejs-portal-sitel-www-2             2/2      Running   0           4m9s   888.16.142.217  apimdev0129  <none>         <none>
ibm-apiconnect-75b47f9f87-p25dd    1/1      Running   0           5m12s  888.16.109.207  apimdev0103  <none>         <none>
```

The upgrade to version 10.0.3.0 is started and the following pod list is observed:

```
$ kubectl get po -o wide
NAME          READY    STATUS    RESTARTS   AGE   IP             NODE          NOMINATED NODE
READINESS GATES
ejs-portal-nginx-84f57ffd8c-hbf66  1/1      Running   0           10m    888.16.109.208  apimdev0103  <none>         <none>
ejs-portal-nginx-84f57ffd8c-mvq96  1/1      Running   0           10m    888.16.142.215  apimdev0129  <none>         <none>
ejs-portal-nginx-84f57ffd8c-vpmtl  1/1      Running   0           10m    888.16.142.214  apimdev0129  <none>         <none>
ejs-portal-sitel-db-0              2/2      Running   0           10m    888.16.109.209  apimdev0103  <none>         <none>
ejs-portal-sitel-db-1              0/2      Pending   0           91s    <none>         <none>         <none>
ejs-portal-sitel-db-2              2/2      Running   0           2m41s  888.16.142.218  apimdev0129  <none>         <none>
```


ejs-portal-sitel-www-0	2/2	Running	0	9m51s	888.16.109.210	apimdev0103	<none>	<none>
ejs-portal-sitel-www-1	2/2	Running	0	12m	888.16.142.213	apimdev0129	<none>	<none>
ejs-portal-sitel-www-2	0/2	Pending	0	111s	<none>	<none>	<none>	<none>
ibm-apiconnect-75b47f9f87-p25dd	1/1	Running	0	10m	888.16.109.207	apimdev0103	<none>	<none>

The pod list shows that **db-2** has restarted, and has been rescheduled to **apimdev0129** as there were no other db pods running on that node. However, **db-1** and **www-2** are both stuck in **Pending** state, as there is already a pod of the same type running on the worker node that is hosting the local storage that they are bound to. If you run a **describe** command of either pod you will see the following output:

```
Events:
  Type            Reason              Age             From              Message
  ----            -
  Warning         FailedScheduling   10s (x4 over 2m59s)  default-scheduler  0/3 nodes are available: 1 node(s) didn't match pod affinity/anti-affinity, 1 node(s) didn't match pod anti-affinity rules, 2 node(s) had volume node affinity conflict.
```

To resolve this situation you need to delete the PVCs for each pod, and then delete the pod itself, so that Kubernetes will regenerate the PVCs and schedule the pod on the worker node that does not have the anti-affinity conflict.

Therefore, for the **db-1** pod the following commands must be run:

```
$ kubectl get pvc | grep ej-s-portal-sitel-db-1
db-ejs-portal-sitel-db-1          Bound          local-pv-fa445e30    250Gi          RWO            local-storage    15m
dblogs-ejs-portal-sitel-db-1    Bound          local-pv-d57910e7    250Gi          RWO            local-storage    15m
```

```
$ kubectl delete pvc db-ejs-portal-sitel-db-1 dblogs-ejs-portal-sitel-db-1
persistentvolumeclaim "db-ejs-portal-sitel-db-1" deleted
persistentvolumeclaim "dblogs-ejs-portal-sitel-db-1" deleted
```

```
$ kubectl delete po ej-s-portal-sitel-db-1
pod "ej-s-portal-sitel-db-1" deleted
```

For the **www-2** pod the following commands must be run:

```
$ kubectl get pvc | grep ej-s-portal-sitel-www-2
admin-ejs-portal-sitel-www-2    Bound          local-pv-48799536    245Gi          RWO            local-storage    51m
backup-ejs-portal-sitel-www-2   Bound          local-pv-a93f5607    245Gi          RWO            local-storage    51m
web-ejs-portal-sitel-www-2      Bound          local-pv-facd4489    245Gi          RWO            local-storage    51m
```

```
$ kubectl delete pvc admin-ejs-portal-sitel-www-2 backup-ejs-portal-sitel-www-2 web-ejs-portal-sitel-www-2
persistentvolumeclaim "admin-ejs-portal-sitel-www-2" deleted
persistentvolumeclaim "backup-ejs-portal-sitel-www-2" deleted
persistentvolumeclaim "web-ejs-portal-sitel-www-2" deleted
```

```
$ kubectl delete po ej-s-portal-sitel-www-2
pod "ej-s-portal-sitel-www-2" deleted
```

If the PVC has **persistentVolumeReclaimPolicy: Delete** set on it, as is the case for the OVA deployments, then no cleanup is necessary as the old data will have been deleted on the worker node that is no longer running the **db-1** and **www-2** pods.

Kubernetes can now reschedule the pods. All pods with persistent storage are now spread across the available worker nodes, and the pods whose PVCs were deleted will get a full copy of the data from the existing running pods. The following pod list is now observed in our example:

```
$ kubectl get po -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE                                NOMINATED NODE
READINESS GATES
ej-s-portal-nginx-84f57ffd8c-f85wm  1/1    Running   0           30s   888.16.29.136   apimdev1066                        <none>         <none>
ej-s-portal-nginx-84f57ffd8c-k5klb  1/1    Running   0           103s  888.16.142.220   apimdev0129                        <none>         <none>
ej-s-portal-nginx-84f57ffd8c-lqhqs  1/1    Running   0           1m53s 888.16.109.212   apimdev0103                        <none>         <none>
ej-s-portal-sitel-db-0              2/2    Running   0           6m43s 888.16.109.211   apimdev0103                        <none>         <none>
ej-s-portal-sitel-db-1              2/2    Running   0           8m20s 888.16.29.134   apimdev1066                        <none>         <none>
ej-s-portal-sitel-db-2              2/2    Running   0           14m   888.16.142.218   apimdev0129                        <none>         <none>
ej-s-portal-sitel-www-0             2/2    Running   0           93s   888.16.109.213   apimdev0103                        <none>         <none>
ej-s-portal-sitel-www-1             2/2    Running   0           3m55s 888.16.142.219   apimdev0129                        <none>         <none>
ej-s-portal-sitel-www-2             2/2    Running   0           7m27s 888.16.29.135   apimdev1066                        <none>         <none>
ibm-apiconnect-75b47f9f87-p25dd    1/1    Running   0           22m   888.16.109.207   apimdev0103                        <none>         <none>
```

Issues when installing Drupal 8 based custom modules or sub-themes into the Drupal 9 based Developer Portal

From IBM API Connect 10.0.3.0, the Developer Portal is based on the Drupal 9 content management system. If you want to install Drupal 8 custom modules or sub-themes into the Drupal 9 based Developer Portal, you must ensure that they are compatible with Drupal 9, including any custom code that they contain, and not using any deprecated APIs, for example. There are tools available for checking your custom code, such as [Drupal check](#) on GitHub, which checks Drupal code for deprecations.

For example, any Developer Portal sites that contain modules or sub-themes that don't contain a Drupal 9 version declaration will fail to upgrade, and errors like the following output will be seen in the **admin** logs:

```
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: Checking theme: emeraldgreen
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '' found for emeraldgreen
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: Checking theme: rubyred
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '8.x' found for rubyred
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: Found themes incompatible with
Drupal 9: emeraldgreen rubyred
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: /tmp/restore_site.355ec8 is NOT
Drupal 9 compatible
...
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: Checking module: custom_mod_1
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '' found for custom_mod_1
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: Checking module: custom_mod_2
```

```
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '8.x' found for custom_mod 2
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: Found modules incompatible with
Drupal 9: emeraldgreen rubyred
[ queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: site1.com is NOT Drupal 9
compatible
```

To fix version compatibility errors, all custom modules and sub-themes should declare a `core_version_requirement` key in their `*.info.yml` file that indicates Drupal 9 compatibility. For example:

```
name: Example module
type: module
description: Purely an example
core: 8.x
core_version_requirement: '^8 || ^9'
package: Example module
```

```
# Information added by Drupal.org packaging script on 2020-05-31
version: '8.x-1.3'
project: 'example_module'
datestamp: 1590905415
```

This example specifies that the module is compatible with all versions of Drupal 8 and 9. For more information, see [Let Drupal know about your module with an .info.yml file](#) on the drupal.org website.

If you have a backup of a site that you need to restore, and are getting the version compatibility error, but the module or theme `*.info.yml` file cannot be changed easily, then you have two options. Either:

- Add an environment variable into the portal CR for the `www` pod of the `admin` container stating `SKIP_D9_COMPAT_CHECK: "true"`. However, if you choose this method, you must be positive that all of the custom modules and themes for your sites are Drupal 9 compatible, as otherwise the sites may end up inaccessible after the upgrade or restore.
 1. On VMware, create an extra values file to contain the environment variable, as follows:

```
spec:
  template:
    - containers:
      - env:
        - name: SKIP_D9_COMPAT_CHECK
          value: "true"
        name: admin
      name: www
```

2. Save the file as `d9compat.yaml`, and run the following command:

```
apicup subsys set <portal_subsystem_name> extra-values-file d9compat.yaml
```

3. Then, update the portal VMware with the updated setting by running the following command:

```
apicup subsys install <portal_subsystem_name>
```

Or:

- Extract the site backup, edit the relevant files inside it, and then tar the backup file again. Note that this procedure will overwrite the original backup file, so ensure that you keep a separate copy of the original file before you start the extraction. For example:
 1. `mkdir /tmp/backup`
 2. `cd /tmp/backup`
 3. `tar xzf path_to_backup.tar.gz`
 4. Edit the custom module and theme files to make them Drupal 9 compatible, and add the correct `core_version_requirement` setting.
 5. `rm -f path_to_backup.tar.gz`
 6. `tar cfz path_to_backup.tar.gz`
 7. `cd /`
 8. `rm -rf /tmp/backup`

Skipping health check when re-running upgrade

The `apicup subsys install` command automatically runs `apicup health-check` prior to attempting the upgrade. An error is displayed if a problem is found that will prevent successful upgrade

In some scenarios, if you encounter an upgrade failure, an attempt to rerun `apicup subsys install` is blocked by errors found by `apicup health-check`. Even when you have fixed the error (such as reconfiguration of an incorrect upgrade CR), the failed upgrade can continue to cause the health check to fail.

You can work around the problem by adding the `--skip-health-check` flag to suppress the health check:

```
apicup subsys install <subsystem_name> --skip-health-check
```

In this case, use of `--skip-health-check` allows the upgrade to rerun successfully.

Maintaining API Connect

You can use utilities to complete maintenance tasks such as backup, restore, and certificate management in a VMware environment.

Note: When maintaining API Connect, do not use `kubectl exec` commands to access API Connect pods unless advised by IBM.

Do not make any changes on the deployed VMs unless documented here or otherwise advised by IBM. Attempting to manually update packages, adding new users, or installing new software will likely cause problems. Operating system updates are handled by API Connect fix packs.

- [Advanced installation on VMware](#)
Optionally install additional software.
- [Advanced configuration on VMware](#)
Perform configuration tasks on your API Connect deployment on VMware.
- [Advanced configuration for management subsystem](#)
Specify advanced configuration for the management subsystem.
- [Backing up and restoring on VMware](#)
Backup and restore API Connect on VMware.
- [Disaster recovery](#)
Prepare for disaster events and recover API Connect on VMware if a disaster event occurs.
- [Disabling and re-enabling the Analytics subsystem on VMware](#)
Disable the Analytics subsystem by shutting down the VM that hosts it.
- [Maintaining a two data center deployment](#)
How to maintain a two data center disaster recovery (DR) deployment on VMware, including information about normal operation, failure handling, and recovery.
- [Monitoring and health checks](#)
Monitor your deployment and complete health checks on subsystems.
- [Logging](#)
Configure logging for your deployment on VMware.
- [Troubleshooting](#)
Troubleshoot your deployment on VMware.

Advanced installation on VMware

Optionally install additional software.

See:

- [Installing the IBM License Metric Tool on VMware](#)
The IBM License Metric Tool (ILMT) helps you determine whether your IBM® API Connect deployment is compliant with licensing requirements.
- [Installing the toolkit](#)
- [Installing the Automated API behavior testing application](#)
Install the Automated API behavior testing application in your IBM API Connect deployment on VMware.

Installing the IBM License Metric Tool on VMware

The IBM License Metric Tool (ILMT) helps you determine whether your IBM® API Connect deployment is compliant with licensing requirements.

About this task

>ILMT provides useful features for managing virtualized environments and measuring license utilization. ILMT discovers the software that is installed in your infrastructure, helps you to analyze the consumption data, and allows you to generate audit reports. Each report provides you with different information about your infrastructure, for example the computer groups, software installations, and the content of your software catalog.

By default, every ILMT audit report presents data from the previous 90 days. You can customize the type and amount of information displayed in a report by using filters, and save your personal settings for future use. You can also export the reports to .csv or .pdf format, and schedule report emails so that specified recipients are notified when important events occur.

For more information about ILMT, see the [IBM License Metric Tool](#) documentation.

Note: The following instructions describe the ILMT installation for API Connect deployments in a VMware environment, and apply to the Management, Developer Portal, and Analytics OVAs.

Procedure

To download the BigFix Agent for the Ubuntu operating system, complete the following steps:

1. Navigate to <http://support.bigfix.com/bes/release/>, and then follow the link to the latest release; for example: <http://support.bigfix.com/bes/release/9.5/patch11/>.
2. In the Agent section of the table, click the Download link corresponding to the following release:
 - Operating System: Ubuntu
 - Version: 16
 - Architecture: x86_64

In this example, the direct download link is: <http://software.bigfix.com/download/bes/95/BESAgent-9.5.11.191-ubuntu10.amd64.deb>

Upload the BigFix Agent package to each OVA server, as follows:

3. Upload the BigFix Agent .deb package to each running OVA server. Following is an example of the command for uploading the file that was downloaded in Step 2:

```
scp BESAgent-9.5.11.191-ubuntu10.amd64.deb apicadm@<OVA_Hostname>:~
```

The generic command is:

```
scp <BigFix_Agent_Package_File_Path> apicadm@<OVA_Hostname>:~
```

The BigFix Agent .deb package will be uploaded to `/home/apicadm`.

Install the BigFix client on each OVA server, as follows:

4. Log in, with a secure shell connection, to the server as the `apicadm` user.

5. Become root by entering `sudo -s`.
6. Follow [IBM Bigfix installation instructions Ubuntu](#) to install the .deb package.

What to do next

Ensure that TCP/IP port 52311 is open on your firewall.

Installing the toolkit

You can install the toolkit that provides CLI commands, and the API Designer user interface, for IBM® API Connect.

About this task

The toolkit is provided as executable files, so no actual installation is necessary, you just need to download the required compressed file and extract the contents.

There are two toolkit options available:

- CLI: provides a command line environment for working with IBM API Connect.
- CLI + LoopBack + Designer: provides a command line environment for working with IBM API Connect, including LoopBack® support, and the API Designer application.

To install the toolkit, download the compressed file that is appropriate for your chosen toolkit option and platform, then extract the contents to a chosen location on your local machine. The compressed file contains an executable file for running CLI commands and, if you choose the CLI + LoopBack + Designer option, an executable file for launching the API Designer application.

You can download the toolkit from one of the following locations:

- From IBM Fix Central.
- From the Cloud Manager or API Manager user interface.

Procedure

To install and run the toolkit, complete the following steps:

1. Download the toolkit compressed file.
 - To download the toolkit from IBM Fix Central, complete the following steps:
 - Open the [IBM Fix Central site](#) in your browser.
 - In the Product selector field, enter `API Connect`, then select IBM API Connect from the drop down list.
 - Select your Installed Version and click Continue. If you do not know your installed IBM API Connect version, contact your administrator.
 - In the Text field, enter `toolkit`, then click Continue.
 - Select the toolkit file that you want to download.
 - Click Continue, then follow the instructions to complete the download operation.
 - To download the toolkit from Cloud Manager or API Manager user interface, complete the following steps:
 - Open the Cloud Manager or API Manager user interface.
 - On the welcome page, click the Download Toolkit tile. The two toolkit options are listed.
 - Click your platform alongside either the CLI option or the CLI + LoopBack + Designer option as required, then save the associated compressed file to your local file system.
 - To download the API Designer credentials (Client ID and Client Secret), click Download alongside API Designer Credentials, then save the `designer_credentials.json` file to your local file system. A command is provided for installing the credentials, as described in detail in [step 6](#).
 - To download the toolkit credentials (Client ID and Client Secret), click Download alongside Credential, then save the `credentials.json` file to your local file system. A command is provided for installing the credentials, as described in detail in [step 5](#).
 - Close the Install API Connect Toolkit window.

2. Extract the contents of the toolkit compressed file to a folder of your choice.

The contents of the file depend on the your chosen toolkit option and platform, as follows:

- The `apic-slim` or `apic-slim.exe` file is the CLI for IBM API Connect.
- The `apic` or `apic.exe` file is the CLI for IBM API Connect including LoopBack support.
Tip: If you are using the CLI option, then if you rename the `apic-slim` file to `apic`, or the `apic-slim.exe` file to `apic.exe`, you can run the CLI commands exactly as documented, copy and paste sample commands from the documentation, and use any command scripts as-is if you later move to the CLI + LoopBack + Designer option.
- The `api_designer-platform` file is the API Designer user interface application for the specified platform.

3. Optional: Delete the `$HOME/.apiconnect/node_cli` directory.

You need to do this only if you replaced a version of `apic` or `apic.exe` and are using the `apic lb4` command. You need to delete the directory `get apic` to unpack the new loopback.

On Windows, `$HOME` is defined by environment `USERPROFILE`.

4. Run the CLI.

- For the Mac OSX or Linux® platforms, complete the following steps:
 - Open a terminal instance and navigate to the folder where you extracted the contents of the toolkit compressed file.
 - Make the CLI file an executable file by entering the following command:

```
chmod +x download_name
```

Where `download_name` is the name of the toolkit file that you downloaded, either `apic` or `apic-slim`.

- Run CLI commands as follows:

```
./apic command_name_and_parameters
```

or

```
./apic-slim command_name_and_parameters
```

For details of the CLI commands, see [apic](#).

- For the Windows platform, complete the following steps:
 - Open a command prompt and navigate to the folder where you extracted the contents of the toolkit compressed file.
 - Run CLI commands as follows:

```
apic command_name_and_parameters
```

or

```
apic-slim command_name_and_parameters
```

For details of the CLI commands, see [apic](#).

Tip: Add the folder location of your CLI file to your PATH variable so that you can run CLI commands from anywhere in your file system.

5. Install the toolkit credentials.

If you do not install the toolkit credentials that are provided for download, as detailed previously in the [toolkit credentials download instructions](#), API Connect uses a default set of credentials that are identical for all deployments. However, the downloaded credentials were generated during the deployment of your API Connect system and are unique to your installation. To install the credentials into your local toolkit, run the following command:

```
apic client-creds:set toolkit_credentials_file_path/credentials.json
```

where *credentials_file_path* is the location to which you downloaded the toolkit credentials JSON file. After you have run this command, your toolkit uses these new credentials to authenticate with the management server.

Note: At any one time, you can use only one set of toolkit credentials for login to a management server from the toolkit CLI. If you want to log in to a different management server you must install the toolkit credentials from that management server.

To revert to the default toolkit credentials for all login operations from the toolkit CLI, use the following command:

```
apic client-creds:clear
```

For increased security, an administrator can remove the default credentials from the management server by completing the following steps:

- a. Log in to the management server as an administrator; see [Logging in to a management server](#).
- b. Delete the default credentials by running the following commands:

```
apic registrations:delete toolkit --server mgmt_endpoint_url  
apic registrations:delete consumer-toolkit --server mgmt_endpoint_url
```

6. Install the API Designer credentials.

If you do not install the API Designer credentials that are provided for download, as detailed previously in the [API Designer credentials download instructions](#), API Connect uses a default set of credentials that are identical for all deployments. However, the downloaded credentials were generated during the deployment of your API Connect system and are unique to your installation. To install the custom credentials into your local API Designer, set the **APIC_DESIGNER_CREDENTIALS** environment variable to the credentials download location, using the mechanism appropriate for your operating system.

After you have set the **APIC_DESIGNER_CREDENTIALS** environment variable, your API Designer uses the new credentials to authenticate with the management server.

- Windows:

If you are using Windows, create an environment variable using one of the following methods:

- Create a permanent environment variable so that you can start API Designer from any location on your computer:
 - Open the Environment Variables page: Click Start, Settings, System and in the "Related Settings" section of the page, click Advanced System Settings.
 - On the Advanced tab of the System Properties dialog box, click Environment Variables.
 - In the "User variables" section, click New and create an environment variables with the following settings:
 - Variable: **APIC_DESIGNER_CREDENTIALS**
 - Value: *<designer_credentials_file_path>*\designer_credentials.json where *<designer_credentials_file_path>* is the location where you stored the designer_credentials.json file.
 - Click OK to save the new environment variable, and then exit the dialog box.
- If you don't have the required permissions to create a permanent environment variable, you can create a temporary one that will only be used while you are running the API Designer application. The following steps must be performed every time you start API Designer:
 - Open the Windows command prompt.
 - Run the following command to set the temporary environment variable:

```
set APIC_DESIGNER_CREDENTIALS=<designer_credentials_file_path>\designer_credentials.json
```

where *<designer_credentials_file_path>* is the location where you stored the designer_credentials.json file.

Note: Leave the command prompt open for the next step. You must set the temporary variable and start the API Designer within the same Windows command session.

- Now run the following command to start API Designer:

```
C:\>"Program Files\API Designer\API Designer.exe"
```

By default, API Designer is installed in the C:\Program Files\API Designer folder as shown in the example. If you installed it into a different location, use your own location in the command. Note that the path and file name are enclosed in quotation marks because they contain spaces.

- Mac:

If you are using Mac OS X, create an environment variable using one of the following methods:

- Run the following command to set the global environment variable:

```
launchctl setenv APIC_DESIGNER_CREDENTIALS <designer_credentials_file_path>/designer_credentials.json
```

where *<designer_credentials_file_path>* is the location where you stored the API Designer credentials JSON file.

- Pass in the environment variable while starting API Designer from the command line. With this method, you must run the following command every time you start API Designer:

```
APIC_DESIGNER_CREDENTIALS=<designer_credentials_file_path>/designer_credentials.json open
<designer_application_file_path>/'API Designer.app'
```

where:

- `<designer_credentials_file_path>` is the location to which you downloaded the API Designer credentials JSON file.
- `<designer_application_file_path>` is the location to which you downloaded and uncompressed the API Designer application.

7. Start the API Designer application from the location to where you stored the extracted file.

Note:

- To uninstall the API Designer application on a Windows platform with a non Administrator account, complete the following steps:
 - In Windows File Explorer, navigate to the `USER_HOME\AppData\Local\Programs\api-designer` folder.
 - Run the **Uninstall API Designer application** application. Do **not** use the **Add or remove programs** window.
- To uninstall the API Designer application on a Windows platform with an Administrator account, you can either run the **Uninstall API Designer application** application, or you can use the **Add or remove programs** window.

Results

The IBM API Connect toolkit CLI and, if selected, the API Designer user interface application are installed on your local system.

For information on using the API Designer user interface, see [Developing your APIs and applications](#).

For information on using the toolkit CLI, see [Using the developer toolkit command-line tool](#).

Installing the Automated API behavior testing application

Install the Automated API behavior testing application in your IBM® API Connect deployment on VMware.

About this task

Use the following steps to install the Automated API behavior testing application as part of your initial deployment, or to add it in later.

Note: It is not possible to use the Automated API behavior testing application with a two data center disaster recovery configuration ([A two data center deployment strategy on VMware](#)).

Procedure

1. Run the following `apicup` commands to install or update your deployment with the the Automated API behavior testing application:

```
apicup subsys set mgmt test-and-monitor-enabled true
apicup certs set mgmt hub --clear
apicup certs set mgmt turnstile --clear
apicup subsys set mgmt hub hub-endpoint
apicup subsys set mgmt turnstile turnstile-endpoint
apicup subsys install mgmt
```

where:

- `mgmt` is the name of your Management server (which you are either installing or updating).
- `hub-endpoint` is an endpoint on the ingress subdomain for the API Connect stack; for example: `hub.testsrv0231.subnet1.example.com`.
- `turnstile-endpoint` is an endpoint on the ingress subdomain for the API Connect stack; for example: `turnstile.testsrv0231.subnet1.example.com`.

Attention: The hub and turnstile endpoints should be distinct and must not overlap with any of the 4 management endpoints.

2. Verify that the application deployed properly by completing the following steps:

- a. Log in to API Manager.
- b. On the Home page, click the Test APIs tile.
- c. Verify that the Automated API behavior testing opens.

Advanced configuration on VMware

Perform configuration tasks on your API Connect deployment on VMware.

See:

- [Adding a static route on a virtual machine](#)
You can add a static route to the routing table when deploying API Connect on a VMware virtual machine.
- [Adding disk space to a VMware appliance](#)
Increase disk space on the VMware appliance.
- [Appliance boot mode configuration](#)
Configure the boot mode for the appliance VMs.
- [Running commands at boot with iso-ignored boot mode](#)
When `iso-ignored` boot mode is configured, complete the following steps on each appliance that should run commands at boot.
- [Changing deployment profiles on VMware](#)
You can change an API Connect installation to use a different deployment profile.
- [Changing the DNS server configuration on appliances](#)
Complete this task only if you must change the DNS addresses after the initial boot of the VM.
- [Configuring API Connect subsystems in a cluster](#)
This topic describes how to configure a cluster of API Connect subsystems (management server, analytics, and Developer Portal) with three VMs for each

- subsystem, for use with a load balancer, to support a high availability (HA) environment.
- [Configuring SSHD to limit access to your deployment](#)
How to configure SSHD in order to limit host access to your VMware deployment.
- [Enabling Developer Portal feature flags on VMware](#)
Use a template override to update the installation CR with settings that control the default behavior of the Developer Portal subsystem.
- [Managing an appliance data disk](#)
You can use `apic` commands to manage appliance data disks in your VMware deployment.
- [Replica override for microservices on VMware](#)
Modify the replica count for microservices by using template overrides for Custom Resources.
- [Testing a new configuration against a current cluster](#)
You can use the `apicup` utility to test the validity of a new configuration before you install it.
- [Use JWT security instead of mTLS between subsystems](#)
If your network requires TLS termination on load-balancers located between your API Connect subsystems, then disable mTLS and enable JWT to provide application layer security.

Adding a static route on a virtual machine

You can add a static route to the routing table when deploying API Connect on a VMware virtual machine.

About this task

Add commands to your additional `cloud-init` file. You can do this for the Management, Analytics, and Portal subsystems.

Procedure

1. Specify an additional cloud-init file:

```
apicup subsys set <subsys> additional-cloud-init-file <path-to-cloud-init-file>
```

2. Add lines to your cloud-init file.

Syntax:

```
bootcmd:  
- ip route add <destination>/<mask> via <gateway> dev eth<n>
```

Example:

```
bootcmd:  
- ip route add 172.27.218.0/24 via 172.27.218.1 dev eth1  
- ip route add 172.26.203.0/24 via 172.27.218.1 dev eth1
```

3. Regenerate the ISO file.

```
apicup subsys install <subsys> --out <plan-directory>
```

4. Restart the ISO, and ensure that the node(s) have this updated ISO attached at startup.

Adding disk space to a VMware appliance

Increase disk space on the VMware appliance.

About this task

For multi-node deployment of the appliance, only one VM at a time should have its disk space augmented. The subsystem should be in good health before proceeding to the next node. Health of the subsystem can be checked with `apicup subsys health-check <subsystem-name>`.

Restriction:

The increased disk capacity of a VMware appliance won't be reflected in the storage size of the PVCs that are hosted on the appliance, as those PVCs are immutable. This restriction doesn't impact the Portal and Analytics subsystems, as the PVC storage size isn't used to enforce any disk usage limit, so the subsystems can use all of the additional disk space. However, for the Management subsystem the PVC storage size is used in the database storage utilization computation, and that might trigger disk usage warnings.

The most common reason for the Management subsystem database to run out of disk space, is due to Write Ahead Log (WAL) file accumulation in relation to an incorrectly configured backup, or an unavailable backup server. If this type of condition arises, adding disk space might help with the health of the appliance control plane, but it won't increase the disk space that's available to the database, so you must resolve the WAL file accumulation condition. If the disk space that's available to the Management database must be increased, you should carry out a disaster recovery operation of the Management subsystem, using a larger data disk size for the new installation, so that adding disk space after the install won't be needed.

Procedure

1. Shutdown the virtual machine, either by issuing a shutdown command as root from the VM, or by using the VMware console or command line interface.
For example, use the VMware console user interface and click the red rectangle.
2. Use either the VMware console or command line interface to add a new device to the VM, of type Hard Disk and of the desired size to be added to the appliance.
For example, on the VMware console:
 - a. Click Edit settings.

- b. Click Add new device. From the drop-down menu, select Hard Disk.
- c. Enter size in GB and click OK.
3. Use either the VMware console or the command line interface to power on the VM.
As the VM starts it should detect the newly added disk and use it to augment the available capacity for the /data/secure mount point.
4. Optionally you may verify that a new disk `sdc` was added (or `sdd` if you already added one disk, and so on). Log in to the server and observe the new disk. For example:

```
root@apimdev1139:~# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda         8:0    0  100G  0 disk
├─sda1      8:1    0  99.9G  0 part /
├─sda14     8:14   0    4M  0 part
└─sda15     8:15   0  106M  0 part /boot/efi
sdb         8:16   0  250G  0 disk
├─apicconnect-data 252:0   0  260G  0 lvm
└─apicSecureDisk 252:1   0  260G  0 crypt /data/secure
sdc         8:32   0   10G  0 disk
├─apicconnect-data 252:0   0  260G  0 lvm
└─apicSecureDisk 252:1   0  260G  0 crypt /data/secure
sr0        11:0    1   44K  0 rom
```

Appliance boot mode configuration

Configure the boot mode for the appliance VMs.

By default every appliance node requires the presence of an ISO file attached at boot.

Version 10.0.4.0 or later: You can configure individual VMs so that they can be booted and rebooted without an ISO being attached. Each appliance VM (node) is configured individually.

You can:

- [Check the current boot mode of an appliance](#)
- [Configure an appliance for iso-ignored boot mode](#)
- [Configure an appliance for iso-required boot mode](#)
- [Revert back to iso-required boot mode by booting the appliance in rescue mode](#)

Check the current boot mode of an appliance

The default boot mode for all appliances is `iso-required`. To check the current boot mode:

1. Connect to the target appliance via SSH then switch to the root user:

```
ssh apicadm@{ova appliance hostname}
sudo -i
```

2. Use `apic boot-mode` to check the current boot mode of the appliance. For example:

```
root@ip-192-168-122-241:~# apic boot-mode
INFO[0000] Log level: info
INFO[0000] current boot mode is: iso-required
```

Configure an appliance for iso-ignored boot mode

An appliance can be configured to use the `iso-ignored` boot mode instead of the default `iso-required`. Prerequisite: The appliance must have first booted with its configuration ISO attached and in `iso-required` boot mode (default) before the boot mode can be changed to `iso-ignored`.

You can configure an appliance to use the `iso-ignored` boot mode instead of the default `iso-required`.

Prerequisite: The appliance must have first booted with its configuration ISO attached and in `iso-required` boot mode (default), before you can change the boot mode to `iso-ignored`.

Note: If you added a static route to the routing table, as documented in [Adding a static route on a virtual machine](#), and if you are relying on a `bootcmd` from the ISO, then using the `iso-ignored` boot mode causes the `bootcmd` section being ignored or ineffective.

1. Connect to the target appliance via SSH then switch to the root user:

```
ssh apicadm@{ova appliance hostname}
sudo -i
```

2. Edit the file `/etc/default/grub.d/50-cloudimg-settings.cfg` to ensure that the `GRUB_TIMEOUT` and `GRUB_RECORDFAIL_TIMEOUT` are set to the value 5. Then run the command `update-grub`. For example:

```
root@ip-192-168-122-241:~# update-grub
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/50-cloudimg-settings.cfg'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.4.0-84-generic
Found initrd image: /boot/initrd.img-5.4.0-84-generic
done
```

3. Set the appliance boot mode to `iso-ignored` with the command `apic boot-mode iso-ignored`. For example:


```
root@ip-192-168-122-241:~# apic boot-mode iso-ignored
INFO[0000] Log level: info
INFO[0000] current boot mode is: iso-ignored
```

- Shutdown the VM, and remove the CD/DVD device from the VM configuration through VSphere. For any subsequent boot of the appliance while it remains in `iso-ignored` boot mode, any ISO file attached will be ignored, and the boot will proceed as normal, whether an ISO file is attached or not.

Warning: If the appliance was booted in `iso-ignored` boot mode, it is possible to change its boot mode to `iso-required`. However, as stated in the prerequisites, it is not possible to immediately switch back to `iso-ignored`, and the next boot of the appliance must happen with the configuration ISO attached and boot mode `iso-required`.

Configure an appliance for `iso-required` boot mode

The mode `iso-required` is the default boot mode for all appliances. If the appliance boot mode was changed to `iso-ignored` you can revert back to `iso-required` boot mode.

Warning:

If the appliance was booted in `iso-ignored` boot mode, it is possible to change its boot mode to `iso-required`. However, you cannot immediately switch back to `iso-ignored`, and the next boot of the appliance must happen with the configuration ISO attached and boot mode `iso-required`. After the appliance is booted with ISO attached in `iso-required` boot mode, the boot mode can then be switched to `iso-ignored` if needed.

- Connect to the target appliance via SSH then switch to the root user:

```
ssh apicadm@{ova appliance hostname}
sudo -i
```

- Set the appliance boot mode to `iso-required` with the command `apic boot-mode iso-required`. For example:

```
root@ip-192-168-122-241:~# apic boot-mode iso-required
INFO[0000] Log level: info
INFO[0000] current boot mode is: iso-required
```

Important: For any subsequent boot of the appliance while it remains in `iso-required` boot mode, the configuration ISO file must be connected at boot and for every boot.

Revert back to `iso-required` boot mode by booting the appliance in rescue mode

When an appliance is configured with the `iso-ignored` boot mode, the boot sequence ignores any attached ISO. This means that it is not possible to influence the appliance configuration during its boot. In case the appliance is not booting properly and the boot mode needs to be reverted back to `iso-required`, proceed as follows:

- Access to the VMware console for the Appliance, and after causing it to reboot, bring up a console to the appliance. You can give focus to the window by clicking the mouse into it. Press the `Esc` key to access the Linux grub boot menu. For example:

```
GNU GRUB version 2.04
```

```
*Ubuntu
Advanced options for Ubuntu
```

- Press `e` and navigate using arrow keys to the line starting with `linux` to append `systemd.unit=rescue.target`

```
GNU GRUB version 2.04
.
.
.
insmod part_gpt
insmod ext2
if [ x$feature_platform_search_hint = xy ]; then
  search --no-floppy --fs-uuid --set=root e46dalb8-55ao-4df9-842e-\
8d80c3a22ffc
else
  search --no-floppy --fs-uuid --set=root e46dalb8-55ao-4df9-842e-8\
d80c3a22ffc
fi
linux /boot/vmlinuz-5.4.0-84-generic root=LABEL=cloudimg-roc\
tfs ro console=tty1 console=ttyS0 net.ifnames=0 biosdevname=0 systemd.unit\
=rescue.target
initrd /boot/initrd.img-5.4.0-84-generic
```

- Press `Ctrl+x` to boot.
- Press `Enter` to enter maintenance mode. Run the commands:

```
rm -f /etc/cloud/cloud-init.disabled
```

and

```
echo -n "iso-required" > /usr/local/lib/appliance-config/boot-mode
```

```
[ 3.442687] systemd[1]: Mounted Kernel Debug Filesystem
[ 3.444253] systemd[1]: Mounted Kernel Trace Filesystem
[ 3.445680] systemd[1]: Finished Create list of static device nodes for the current kernel
[ 3.446777] systemd[1]: Finished Set console scheme.
[ 3.449347] EXT4-fs (sda1): re-mounted. Opts: (null)
[ 3.452687] systemd[1]: Finished Remount Root and Kernel File Systems
[ 3.455123] systemd[1]: Finished Load Kernel Modules.
[ 3.458487] systemd[1]: Mounting FUSE Control File System...
[ 3.461333] systemd[1]: Mounting Kernel Configuration File System...
```

```
[ 3.462687] systemd[1]: Condition check resulted in Rebuild Hardware Database being skipped.
[ 3.464498] systemd[1]: Condition check resulted in Platform Persistent Storage Archival being skipped.
[ 3.466292] systemd[1]: Starting Load/Save Random Seed...
[ 3.467324] systemd[1]: Starting Apply Kernel Variables...
[ 3.470531] systemd[1]: Starting Create System Users...
[ 3.471847] systemd[1]: Mounted FUSE Control File System
[ 3.472964] systemd[1]: Started Journal Service
Press Enter for maintenance:
(or press Ctrl-D to continue):
root@ip-192-168-122-241:~# rm -f /etc/cloud/cloud-init.disabled
root@ip-192-168-122-241:~# echo -n "iso-required" > /usr/local/lib/appliance-config/boot-mode
root@ip-192-168-122-241:~#
```

5. Reboot the appliance with correct configuration ISO attached.

Running commands at boot with `iso-ignored` boot mode

When `iso-ignored` boot mode is configured, complete the following steps on each appliance that should run commands at boot.

About this task

If you configured the appliance to boot in `iso-ignored` mode as documented in [Appliance boot mode configuration](#), the `bootcmd` lines will be ignored. For each appliance node that should run commands at boot, complete the following steps.

Procedure

1. SSH into the appliance as root.
2. Create the `/usr/local/bin/appliance-boot-commands.sh` script to be executed at boot; for example:

```
cat << EOF > /usr/local/bin/appliance-boot-commands.sh
#!/bin/bash

ip route add 172.27.218.0/24 via 192.168.122.1 dev eth0
ip route add 172.26.203.0/24 via 192.168.122.1 dev eth0
EOF
```

This step takes the same example, and essentially achieves the same result, as the procedure in [Adding a static route on a virtual machine](#).

3. Make the script executable by running the following command:

```
chmod +x /usr/local/bin/appliance-boot-commands.sh
```

4. Create the following systemd service file `/lib/systemd/system/appliance-boot-commands.service`:

```
cat << EOF > /lib/systemd/system/appliance-boot-commands.service
[Unit]
After=network-online.target

[Service]
Type=oneshot
ExecStart=/usr/local/bin/appliance-boot-commands.sh
RemainAfterExit=false
StandardOutput=journal

[Install]
WantedBy=multi-user.target
EOF
```

5. Enable the systemd service by running the following command:

```
systemctl enable appliance-boot-commands
```

Upon subsequent boots, the `/usr/local/bin/appliance-boot-command.sh` script will be executed.

Changing deployment profiles on VMware

You can change an API Connect installation to use a different deployment profile.

For information on the available deployment profiles, see [Planning your deployment topology and profiles](#). The steps for changing deployment profile after installation depend on whether you want to change to a profile that has a different number of VMs and whether additional CPU and memory resources are required for the profile you move to.

Note: The procedures to change deployment profile that are described in this topic do not permit the endpoints to be changed. The API Connect endpoints are listed here: [Deployment overview for endpoints and certificates](#). If you want to change a one replica deployment to a three replica deployment, and you used your VM hostnames in your endpoints, then you must configure load-balancers and logical DNS records so that your original VM hostnames route to your new VMs. Alternatively, you can change endpoints by following the form factor migration procedure: [Migrating from v10 to v10 on a different form factor](#).

Moving to a profile that has the same number of VMs

If you are moving to a profile that requires greater CPU and memory resources, follow these steps:

1. Complete the disaster recovery preparation steps for your subsystems: [Preparing for a disaster](#).

2. Shutdown your API Connect VMs.
3. Update the CPU and memory resources for your API Connect VMs in your VMware console, as required for your new profile.
4. Restart your VMs.
5. In your project directory, update your deployment profile:

```
apicup subsys set <subsystem_name> deployment-profile=<new profile>
```

6. Apply the new profile:

```
apicup subsys install <subsystem_name>
```

If you are moving to a profile that requires lower CPU and memory resources, follow these steps:

1. Complete the disaster recovery preparation steps for your subsystems: [Preparing for a disaster](#).
2. In your project directory, update your deployment profile:

```
apicup subsys set <subsystem_name> deployment-profile=<new profile>
```

3. Apply the new profile:

```
apicup subsys install <subsystem_name>
```

4. Shutdown your API Connect VMs.
5. Update the CPU and memory resources for your API Connect VMs in your VMware console, as required for your new profile.
6. Restart your VMs.

If for any reason you are not able to change the CPU or memory on your VMs, redeploy them following the disaster recovery process: [Recovering from a disaster](#).

Moving from a one replica to a three replica deployment profile

1. Complete the disaster recovery preparation steps for your subsystems: [Preparing for a disaster](#).
Note: You do not need to take VMware snapshots, since your existing VMs are deleted and replaced when you change deployment profile.
2. In your project directory, update your deployment profile:

```
apicup subsys set <subsystem_name> deployment-profile=<three replica profile>
```

3. Define the hostname and network interfaces for the two new VMs in each of your subsystems. Use the `apicup hosts create` and `apicup iface create` commands. For more information, see the hosts and interface create steps in the installation instructions for each subsystem:
 - [Configuring the Management subsystem](#).
 - [Configuring the Analytics subsystem](#).
 - [Configuring the Developer Portal subsystem](#).
4. Shutdown your existing API Connect VMs.
Note: To minimize downtime, you can delay shutting down your existing deployment until you are about to deploy their replacement VMs in the next step.
5. Follow the disaster recovery steps for each subsystem: [Recovering from a disaster](#), but note that you must create new ISO files. Do not attempt to reuse the previous ISO file for one of the VMs in your new three replica deployment.

Moving from a three replica to a one replica deployment profile

1. Complete the disaster recovery preparation steps for your subsystems: [Preparing for a disaster](#).
Note: You do not need to take VMware snapshots, since your existing VMs are deleted and replaced when you change deployment profile.
2. In your project directory, update your deployment profile:

```
apicup subsys set <subsystem_name> deployment-profile=<one replica profile>
```

3. Delete two of your three VMs from your configuration with the `apicup hosts delete` command.
4. Shutdown your existing API Connect VMs.
Note: To minimize downtime, you can delay shutting down your existing deployment until you are about to deploy their replacement VMs in the next step.
5. Follow the disaster recovery steps for each subsystem: [Recovering from a disaster](#), but note that you must create new ISO files. Do not attempt to reuse any previous ISO files for your new one replica deployment.

Changing the DNS server configuration on appliances

Complete this task only if you must change the DNS addresses after the initial boot of the VM.

Before you begin

Verify that you have VMware console access to the VM, as well as the correct password. If you cannot `ssh` into the appliance, you can use the VMware console interactive login instead (use the `apicadm` login and the password configured as `default-password` in the `apicup` project).

About this task

This task is only valid for making changes to DNS resolution. The IP address of the VM cannot be changed at any time after the initial boot.

Procedure

1. Edit the `/etc/netplan/50-cloud-init.yaml` file and update the list of nameserver addresses.

Attention: Typically, you do not edit this file manually. You should only make direct changes to the `/etc/netplan/50-cloud-init.yaml` file when you need to update the nameserver addresses.

The following example shows what the list of nameserver addresses looks like in the file:

```
network:
  ethernets:
    eth0:
      addresses:
        - 192.168.122.242/24
      nameservers:
        addresses:
          - 1.1.1.1
          - 1.0.0.1
          - 8.8.8.8
        search:
          - nip.io
      routes:
        - metric: 10
          to: 0.0.0.0/0
          via: 192.168.122.1
  version: 2
```

2. Run the following command to apply the changes:

```
sudo netplan apply
```

3. Run the following command to verify the changes:

```
cat /run/systemd/resolve/resolv.conf
```

4. Attach an updated ISO to the VM to ensure all artifacts are up-to-date in case a disaster recovery is needed.

- Update the `apicup` project configuration with the correct DNS values.
- Generate a new ISO based on the updated project.
- Attach the new ISO to the VM.

Configuring API Connect subsystems in a cluster

This topic describes how to configure a cluster of API Connect subsystems (management server, analytics, and Developer Portal) with three VMs for each subsystem, for use with a load balancer, to support a high availability (HA) environment.

Before you begin

To create a cluster of hosts for each API Connect subsystem, use `apicup` to create a subsystem with all the required parameters, and to add as many hosts as needed to the configuration file. The configuration file is a `.yaml` file in the project directory.

For each host of a subsystem added in the `.yaml` file, a separate ISO file is created for the cluster member VM. Note that the ISO files for each VM must stay attached during the whole lifetime of the VM.

In an API Connect cluster in a VMware environment, the first three nodes are master nodes. Additional nodes are worker nodes.

Note: To add a new host to an existing cluster, you can create the host, regenerate the ISO and attach it to the new virtual machine, and it will automatically join the cluster.

To use these instructions, you should first review:

- [Requirements for initial deployment on VMware](#)
- [Load balancer configuration in a VMware deployment](#)
- [Deployment overview for endpoints and certificates](#)
- [Firewall requirements on VMware](#)
- [Firewall enabled ports for clustered OVA deployments](#)
- Important: For best performance it is recommended that the network latency between any 2 nodes be as low as possible. Do not configure nodes from the same subsystem cluster across multiple data centers with a high latency network. A high latency network is one that experiences more than 30ms latency between nodes.

About this task

This page presents a concise step-by-step flow, with sample commands, for the configuration of each subsystem. As you step through the flow, you might want to refer back to the detailed configuration steps for each subsystem. The detailed instructions provide additional considerations for each step, and include optional configuration tasks for backups, logging, message queues (analytics), and password hashing. Each of the subsystem pages describe how to use the VMware console to deploy the ISOs that you create here.

Detailed configuration steps:

- [Deploying the Management virtual appliance](#)
- [Deploying the Analytics subsystem](#)
- [Deploying the Developer Portal virtual appliance](#)

The example commands uses the following values for DNS server, internet gateway, host names and IP addresses:

Component	Host names and IP addresses
DNS Name Server	IP: 192.168.1.1
Internet gateway	IP: 192.168.1.2
Manager on VM1	Hostname: manager1.sample.example.com IP: 192.168.1.101

Component	Host names and IP addresses
Manager on VM2	Hostname: <code>manager2.sample.example.com</code> IP: <code>192.168.1.102</code>
Manager on VM3	Hostname: <code>manager3.sample.example.com</code> IP: <code>192.168.1.103</code>
Analytics on VM4	Hostname: <code>analytics1.sample.example.com</code> IP: <code>192.168.1.104</code>
Analytics on VM5	Hostname: <code>analytics2.sample.example.com</code> IP: <code>192.168.1.105</code>
Analytics on VM6	Hostname: <code>analytics3.sample.example.com</code> IP: <code>192.168.1.106</code>
Developer Portal on VM7	Hostname: <code>portal1.sample.example.com</code> IP: <code>192.168.1.107</code>
Developer Portal on VM8	Hostname: <code>portal2.sample.example.com</code> IP: <code>192.168.1.108</code>
Developer Portal on VM9	Hostname: <code>portal3.sample.example.com</code> IP: <code>192.168.1.109</code>

Procedure

1. Create the Management subsystems

a. Create the Management subsystem.

```
apicup subsys create mgmt management
```

b. Set deployment-profile to `n3xc4.m16`.

```
apicup subsys set mgmt deployment-profile=n3xc4.m16
```

If you omit this step, the deployment-profile defaults to `n1xc4.16`, which is for non-HA environments, and will not support three instances of the subsystem in one cluster.

c. Specify the license version you purchased.

```
apicup subsys set mgmt
license-use=<license_type>
```

The `license_type` must be either `production` or `nonproduction`. If not specified, the default value is `nonproduction`.

d. Set management endpoints

Endpoints can point to VM host names, but in cluster deployments typically point to a load balancer. The load balancer distributes requests over the 3 VMs. The following values point to a sample load balancer URL.

Component	Command
Management REST API URL	<code>apicup subsys set mgmt platform-api platform-api.sample.example.com</code>
Consumer (Portal) REST API URL	<code>apicup subsys set mgmt consumer-api consumer-api.sample.example.com</code>
Cloud Manager UI	<code>apicup subsys set mgmt cloud-admin-ui cloud-admin-ui.sample.example.com</code>
API Manager UI	<code>apicup subsys set mgmt api-manager-ui api-manager-ui.sample.example.com</code>

e. Set the search domain for the VM.

```
apicup subsys set mgmt search-domain sample.example.com
```

f. Set the DNS Name Server for the VM to look up endpoints.

```
apicup subsys set mgmt dns-servers 192.168.1.1
```

g. Set a Public Keyfile.

This is the public key of the user account that you want to use to `ssh` from, to the appliance.

```
apicup subsys set mgmt ssh-keyfiles "id_rsa.pub"
```

h. Create the hosts for the subsystem.

You must specify a password to use to encrypt the disks that the appliance uses. Replace the example password in the previous example with a strong password that meets your security requirements.

```
apicup hosts create mgmt manager1.sample.example.com password123
apicup hosts create mgmt manager2.sample.example.com password123
apicup hosts create mgmt manager3.sample.example.com password123
```

i. Set the network interface. Note that the last parameter is the Internet Gateway.

```
apicup iface create mgmt manager1.sample.example.com eth0 192.168.1.101/255.255.255.0 192.168.1.2
apicup iface create mgmt manager2.sample.example.com eth0 192.168.1.102/255.255.255.0 192.168.1.2
apicup iface create mgmt manager3.sample.example.com eth0 192.168.1.103/255.255.255.0 192.168.1.2
```

j. Set the network traffic interfaces.

```
apicup subsys set mgmt traffic-iface eth0
apicup subsys set mgmt public-iface eth0
```

k. Verify the host configuration.

```
apicup hosts list mgmt
```

Note: This command might return the following messages, which you can ignore:

```
* host is missing traffic interface
* host is missing public interface
```

- l. Set a hashed password to access the appliance VM through the VMware Remote Console. Use an operating system utility to create a hashed password, and then use `apicup` to set the hashed password for your subsystem:

```
apicup subsys set mgmt default-password='$1$aTD7uXAO$kNoMAefjGKBwMFiu.8ctr0'
```

Important: Review the requirements for creating and using a hashed password. See [Setting and using a hashed default password](#).

- m. Validate the installation.

```
apicup subsys get mgmt --validate
```

- n. Create an ISO file in a plan folder. For example, `mgmtplan-out`.

```
apicup subsys install mgmt --out mgmtplan-out
```

If you have multiple nodes listed for the hosts, when you run the `--out` command, it creates an ISO for each node. When deploying the nodes from VMware, each node gets its own ISO file attached.

- o. To deploy the management ISOs on VMware, see [Deploying the Management subsystem OVA file](#).

2. Create the analytics subsystems

- a. Create the subsystem:

```
apicup subsys create analyt analytics
```

- b. Set deployment-profile to `n3xc4.m16`.

```
apicup subsys set analyt deployment-profile=n3xc4.m16
```

If you omit this step, the deployment-profile defaults to `n1xc2.16`, which is for non-HA environments, and will not support three instances of the subsystem in one cluster.

- c. Specify the license version you purchased.

```
apicup subsys set analyt
license-use=<license_type>
```

The `license_type` must be either `production` or `nonproduction`. If not specified, the default value is `nonproduction`.

- d. Set analytics ingestion endpoint

Endpoints can point to VM host names, but in cluster deployments typically point to a load balancer. The load balancer distributes requests over the 3 VMs. The following value points to a sample load balancer URL.

Component	Command
analytics-ingestion	<code>apicup subsys set analyt analytics-ingestion=analytics-ingestion.sample.example.com</code>

- e. Set the search domain for the VM.

```
apicup subsys set analyt search-domain sample.example.com
```

- f. Set the DNS Name server for the VM to look up endpoints.

```
apicup subsys set analyt dns-servers 192.168.1.1
```

- g. Set a Public Keyfile.

This is the public key of the user account that you want to use to `ssh` from, to the appliance

```
apicup subsys set analyt ssh-keyfiles "id_rsa.pub"
```

- h. Set a hashed password to access the appliance VM through the VMware Remote Console. Use an operating system utility to create a hashed password, and then use `apicup` to set the hashed password for your subsystem:

```
apicup subsys set analyt default-password='$1$aTD7uXAO$kNoMAefjGKBwMFiu.8ctr0'
```

Important: Review the requirements for creating and using a hashed password. See [Setting and using a hashed default password](#).

- i. Create the hosts for the subsystem. You must specify a password to use to encrypt the disks that the appliance uses. Replace the example password in the following password with a strong password that meets your security requirements.

```
apicup hosts create analyt analytics1.sample.example.com password123
apicup hosts create analyt analytics2.sample.example.com password123
apicup hosts create analyt analytics3.sample.example.com password123
```

- j. Set the network interface.

Note that the last parameter is the Internet Gateway.

```
apicup iface create analyt analytics1.sample.example.com eth0 192.168.1.104/255.255.255.0 192.168.1.2
apicup iface create analyt analytics2.sample.example.com eth0 192.168.1.105/255.255.255.0 192.168.1.2
apicup iface create analyt analytics3.sample.example.com eth0 192.168.1.106/255.255.255.0 192.168.1.2
```

- k. Check the host configuration for problems.

```
apicup hosts list analyt
```

- l. Validate the installation.

```
apicup subsys get analyt --validate
```

- m. Create an ISO file in a plan folder. For example, `analytplan-out`.

```
apicup subsys install analyt --out analytplan-out
```

If you have multiple nodes listed for the hosts, when you run the `--out` command, it creates an ISO for each node. When deploying the nodes from VMware, each node gets its own ISO file attached.

- n. To deploy the ISOs, see [Deploying the Analytics subsystem OVA file](#).

3. Create the portal subsystems

a. Create the portal.

```
apicup subsys create port portal
```

b. For three replica environments, set deployment-profile to `n3xc4.m8`.

```
apicup subsys set port deployment-profile=n3xc4.m8
```

If you omit this step, the deployment-profile defaults to `n1xc4.m8`, which is for non-HA environments, and will not support three instances of the subsystem in one cluster.

c. Specify the license version you purchased.

```
apicup subsys set port  
license-use=<license_type>
```

The `license_type` must be either `production` or `nonproduction`. If not specified, the default value is `nonproduction`.

d. Set the portal endpoints.

Endpoints can point to VM host names, but in cluster deployments typically point to a load balancer. The load balancer distributes requests over the 3 VMs. The following values point to a sample load balancer URL.

Component	Command
portal-admin	<code>apicup subsys set port portal-admin=portal-admin.sample.example.com</code>
portal-www	<code>apicup subsys set port portal-www=portal-www.sample.example.com</code>

e. Set the search domain for the VM.

```
apicup subsys set port search-domain sample.example.com
```

f. Set the DNS Name server for the VM to look up endpoints.

```
apicup subsys set port dns-servers 192.168.1.1
```

g. Set a Public Keyfile.

This is the public key of the user account that you want to use to `ssh` from, to the appliance

```
apicup subsys set port ssh-keyfiles "id_rsa.pub"
```

h. Set a hashed password to access the appliance VM through the VMware Remote Console. Use an operating system utility to create a hashed password, and then use `apicup` to set the hashed password for your subsystem:

```
apicup subsys set port default-password='$1$aTD7uXAO$kNoMAefjGKBwMFiu.8ctr0'
```

Important: Review the requirements for creating and using a hashed password. See [Setting and using a hashed default password](#).

i. Create the hosts for the subsystem. You must specify a password to use to encrypt the disks that the appliance uses.

```
apicup hosts create port portal1.sample.example.com password123  
apicup hosts create port portal2.sample.example.com password123  
apicup hosts create port portal3.sample.example.com password123
```

Replace the example password in the previous example with a strong password that meets your security requirements.

j. Set the network interface.

Note that the last parameter is the Internet Gateway.

```
apicup iface create port portal1.sample.example.com eth0 192.168.1.107/255.255.255.0 192.168.1.2  
apicup iface create port portal2.sample.example.com eth0 192.168.1.108/255.255.255.0 192.168.1.2  
apicup iface create port portal3.sample.example.com eth0 192.168.1.109/255.255.255.0 192.168.1.2
```

k. Check the host configuration for problems.

```
apicup hosts list port
```

l. Validate the installation.

```
apicup subsys get port --validate
```

m. Create an ISO file in a plan folder. For example, `portplan-out`.

```
apicup subsys install port --out portplan-out
```

If you have multiple nodes listed for the hosts, when you run the `--out` command, it creates an ISO for each node. When deploying the nodes from VMware, each node gets its own ISO file attached.

n. To deploy the ISOs, see [Deploying the Developer Portal subsystem OVA file](#).

Configuring SSHD to limit access to your deployment

How to configure SSHD in order to limit host access to your VMware deployment.

About this task

The following example shows how to configure SSHD in order to limit VMware access to a certain set of IP addresses.

Attention:

- SSHD configuration settings are not included in the IBM® API Connect backups. However, the settings are kept when you upgrade to a new fix pack.
- If you change the SSHD cipher level, and encounter issues with the new configuration, you must revert your cipher changes and retest. If after the retest you still have issues, contact IBM Support.

Procedure

1. Log in to the target virtual machine by using an SSH tool, and switch to the root user. For example, to log in to the Management subsystem, run the following commands:

```
ssh ip_address -l apicadm
sudo -i
```

2. Move to the following folder:

```
/etc/ssh/sshd_config.d
```

3. Create a new configuration file with the extension .conf that contains the IP address of the host that you want to allow to access your deployment. For example, to create the file run the following command:

```
vi filename.conf
```

Then, edit the file by using the following format, which in this example is allowing users from IP address 123.135.1.2:

```
AllowUsers *@123.135.1.2
```

Additional hosts can be added by separating the IP address patterns with a space. For example:

```
AllowUsers *@123.135.1.2 *@135.168.1.2
```

4. Restart the SSHD service by running the following command:

```
systemctl restart sshd.service
```

Results

Access to your VMware is now limited to a certain set of IP addresses.

What to do next

If you want to remove a cipher from your SSHD configuration, you can edit the /etc/ssh/sshd_config file. For example, if your security department decides that you must stop using the 128 bit UMAC cipher, you can remove it from your supported ciphers list by editing the /etc/ssh/sshd_config file and changing the following line from:

```
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,umac-128@openssh.com
```

to:

```
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-512,hmac-sha2-256
```

Then restart the SSHD service by running the following command:

```
systemctl restart sshd.service
```

Enabling Developer Portal feature flags on VMware

Use a template override to update the installation CR with settings that control the default behavior of the Developer Portal subsystem.

About this task

You can add a template override to the Developer Portal subsystem CR to change the default behavior of the Portal subsystem, or to turn specific features on and off. Table 1 describes the available feature flags.

Table 1. List of available feature flags for the Developer Portal

Feature flag name	Default Value	Pod	Container	Description
PORTAL_SKIP_WEB_ENDPOINT_VALIDATION	true	www	admin	<p>Controls whether the Portal subsystem checks whether the Portal web endpoint is accessible. Sometimes, the endpoint cannot be reached, for example due to the complexity of the networks. In which case the following type of error can be seen in the <code>portal-www</code> pod, and <code>admin</code> container logs:</p> <pre>An error occurred contacting the provided portal web endpoint: example.com The provided Portal web endpoint example.com returned HTTP status code 504</pre> <p>You can disable the check by setting this flag to <code>false</code>.</p>

Feature flag name	Default Value	Pod	Container	Description
ENABLE_TRUSTED_REVERSE_PROXY	false	www	admin	<p>Controls whether the Drupal <code>trusted_reverse_proxy</code> module can manage the reverse proxies list.</p> <p>If you have multiple load balancers and reverse proxies in your upstream network, and the perimeter module is triggered, the incorrect IP address might be banned rather than the intended client IP address. However, you can manage which IP addresses are banned by using the Drupal <code>trusted_reverse_proxy</code> module, and setting this feature flag to <code>true</code>.</p> <p>The Drupal module inspects the <code>x-forwarded-for</code> headers to identify the reverse proxies, and trusts that the first IP address found in the list is the client IP. For example, in <code>X-Forwarded-For: <client>, <proxy1>, <proxy2></code>, the client IP <code><client></code> will be banned. For more information about this module, see https://www.drupal.org/project/trusted_reverse_proxy.</p> <p>To use this module, you must ensure that all of the proxies in your upstream network, including the ingress controller, are configured correctly so that the <code>x-forwarded-for</code> header is passed through to the <code>portal-www</code> pod. For example, the default behavior of the <code>nginx-ingress-controller</code> is to ignore the inbound <code>x-forwarded-for</code> header, and create a new one. To change this behavior, update the <code>nginx-ingress-controller configmap</code> with the following information:</p> <pre>data: compute-full-forwarded-for: "true" use-forwarded-headers: "true"</pre> <p>Warning: Enable this feature flag only if you trust your upstream reverse proxies, as it is possible to trick the <code>x-forwarded-for</code> header.</p>

Procedure

1. Create an `extra-values.yaml` file, or open an existing `extra-values.yaml` file.
2. In the file, append the feature flag definition to the end of the `spec:` section, making sure to adhere to the spacing used in the file. The following example enables the `trusted_reverse_proxy` feature on all possible UI pages.

```
spec:
  template:
    - containers:
      - env:
        - name: ENABLE_TRUSTED_REVERSE_PROXY
          value: true
          name: container
      name: pod
```

Where:

- `container` is the name of the portal container.
- `pod` is the name of the portal pod.

Note: You can add multiple feature flags to the file.

3. Save the file.
4. Run the following command to set the `extra-values.yaml` file:

```
apicup subsys set <portal_subsystem> extra-values-file <path_to_portal_extra-values.yaml_file>
```

5. Apply the overrides to the portal subsystem by running the following command:

```
apicup subsys install <SUBSYSTEM_NAME>
```

The Developer Portal subsystem is updated to enable or disable features as specified.

Managing an appliance data disk

You can use `apic` commands to manage appliance data disks in your VMware deployment.

The API Connect deployment on VMware uses 2 partitions. The first contains the base operating system, while the second is encrypted so that any customer data that is stored on disk uses the encrypted volume. Encryption is done with Linux Unified Key Setup (LUKS) disk encryption.

During installation of each API Connect subsystem, you specify a password `HD-PASSWD` when configuring the host:

```
apicup hosts create <SUBSYS> <HOSTNAME> <HD-PASSWD>
```

If you want to force a restart of processes, without requiring a full restart of the virtual machine, you can use the command `apic lock` to stop the Kubernetes node on the virtual machine and lock the secured storage. When you are ready to restart processes, you can use `apic unlock` to restart the Kubernetes node. The command `apic unlock` uses the password to unlock the partition so that files can be read from it and written to it. The unlock command also starts the `apic` daemon, also known as the `appliance-manager` service.

You can check that status of this service with:

```
sudo systemctl status appliance-manager
```

One `appliance-manager` daemon runs on each node, where it manages the Kubernetes cluster, processes update and upgrade requests, and gathers logs.

apic command	Description
lock	Shut down appliance services and lock the secured storage
logs	Retrieve logs from all nodes in the cluster
status	Report on cluster status
unlock	Unlock the secure storage and start the appliance
version	Get the API Connect appliance base version

Replica override for microservices on VMware

Modify the replica count for microservices by using template overrides for Custom Resources.

When API Connect is deployed, the deployment profile you select drives the number of replicas to be used for most deployment. For one replica profiles, specified by `n1x` . . . , the replica count is 1. For three replica profiles, specified by `n3x` . . . , the replica count is 3.

You can override the replica count for individual deployments by using template overrides.

- Template overrides are a piece of yaml that should be added to the Kubernetes custom resource. The examples provided in this topic show only that the `template`: element should be under the main `spec`: element of a CR. The rest of the CR is not shown.
- When applying overrides for Appliance deployments (virtual machines), the yaml sample provided as example must be added in an extra-values-file for the subsystem. To view an example of documenting a template override for an Appliance, see [Setting rate limits for public APIs on the management service](#)

Management subsystem

For a Management CR in Kubernetes or OpenShift, the following override specifies 5 replicas for the `apim` deployment:

```
spec:
  template:
    - name: apim
      replicaCount: 5
```

You can also configure replicate count through a similar template override for:

- `juhu`
- `ldap`
- `lur`

Analytics subsystem

The Analytics microservices that can be scaled using template override are:

- `ingestion`
- `mtls-gw`
- `storage-shared`

Note that this microservice cannot be scaled below 2 replicas when using a three replica deployment profile.

Testing a new configuration against a current cluster

You can use the `apicup` utility to test the validity of a new configuration before you install it.

The `apicup` utility is used to set configuration properties for subsystems on API Connect on VMware appliances. The utility can be used for installation, upgrade, and maintenance activities such as backup and restore.

The usage pattern is to set values for each parameter you need for a subsystem, and then deploy the new values with the `apicup subsys install` command.

You can use the `--dry-run` option to test the validity of your values prior to deploying them to an existing cluster:

```
apicup subsys install <subsystem_name> --dry-run
```

Important: You cannot use `apicup subsys install <SUBSYS> --dry-run` for a fresh install or an upgrade. The `--dry-run` feature is for testing proposed configuration changes on an existing cluster.

For example, if you want to configure S3 backups for you management subsystem, the command flow is:

1. Follow the instructions for configuring S3 backups [Reconfiguring backup settings for the management subsystem](#) by issuing the required `apicup subsys set` commands:

```
apicup subsys set mgmt <setting>=<value>
apicup subsys set mgmt <second_setting>=<value>
apicup subsys set mgmt <third_setting>=<value>
.
.
```

2. Do a dry run of the install command:

```
apicup subsys install mgmt --dry-run
```

Example message on success:

```
managementcluster.management.apiconnect.ibm.com/management configured (server dry run)
```

Example message on failure:

```
Error: Failed to execute dry-run:....
```

The error message also contains details about the error.

3. When dry run returns success, deploy:

```
apicup subsys install mgmt
```

Use JWT security instead of mTLS between subsystems

If your network requires TLS termination on load-balancers located between your API Connect subsystems, then disable mTLS and enable JWT to provide application layer security.

JWT security provides application layer security and can be used instead of mTLS when there are load-balancers located between subsystems that require TLS termination. For more information about JWT security, see [Enable JWT instead of mTLS](#).

1. To enable JWT and disable mTLS, first identify the JWKS URL from the management subsystem:

```
apicup subsys get <management subsystem>

...
jwks-url https://appliance1.apic.acme.com/api/cloud/oauth2/certs JWKS URL for Portal and Analytics subsystems to
validate JWT -- this is unsettable and is generated based on the platform-api endpoint
...
```

2. Disable mTLS and enable JWT on portal and analytics by setting the `mtls-validate-client` and `jwks-url` values with `apicup`:

```
apicup subsys set <portal> mtls-validate-client=false
apicup subsys set <portal> jwks-url=appliance1.apic.acme.com/api/cloud/oauth2/certs

apicup subsys set <analytics> mtls-validate-client=false
apicup subsys set <analytics> jwks-url=appliance1.apic.acme.com/api/cloud/oauth2/certs
```

3. Apply the change to portal and analytics with `apicup subsys install`:

```
apicup subsys install <subsystem_name>
```

4. To enable JWT and disable mTLS on the gateway appliance, see *Configuring the API Connect gateway service* in the [DataPower documentation](#).

Note: Do not disable mTLS without enabling JWT.

Advanced configuration for management subsystem

Specify advanced configuration for the management subsystem.

Note: VMware deployments include subsystem settings that cannot be reconfigured without a complete deployment. For example, for the Management Service these settings include hosts, endpoints, interfaces (`public-iface`, `traffic-iface`), IP ranges of the Kubernetes pod and the service networks (`k8s-pod-network`, `k8s-service-network`). Before reconfiguring, review the installation instructions for your subsystem to determine which settings are optional during initial installation and thus able to be updated later.

See:

- [Setting rate limits for public APIs on the management service](#)
Describes the procedure for setting a rate limit for public APIs on the management service. Rate limits provide protection from DDoS (distributed denial of service) attacks.
- [Configuring maximum size of client requests to the Management subsystem on VMware](#)
You can configure the maximum allowed size of the client request body for requests made to the Management subsystem.
- [Configuring monetization on VMware](#)
You can optionally configure monetization in API Connect on a VMware virtual machine by enabling the billing microservice.
- [Configuring use of an external NTP server](#)
You can optionally configure an external NTP server for use by API Connect when deploying on a VMware virtual machine.
- [Overriding resources for Postgres components on VMware](#)
In 10.0.4.0 and greater, you can optionally override the CPU and Memory resources used by postgres database components.
- [Enabling UI feature flags on VMware](#)
Use a template override to update the installation CR with settings that control UI features.

Setting rate limits for public APIs on the management service

Describes the procedure for setting a rate limit for public APIs on the management service. Rate limits provide protection from DDoS (distributed denial of service) attacks.

Before you begin

Note: This article refers to third-party software that IBM does not control. As such, the software may change and this information may become outdated. These instructions assume you have the kubectl command-line tool installed. For more information, see <https://kubernetes.io>.

About this task

Rate limits can be set for public APIs on the management service. Rate limits on APIs help provide protection from DDoS (distributed denial of service) attacks. Without a rate limit, API calls from public APIs are unlimited.

The rate limit configuration requires that the header contains the actual client IP address. Any load balancer or proxy (for example, HAProxy) that is installed in front of the management service must be configured to pass the actual client IP address.

This procedure must be performed on a running API Connect deployment.

Rate limits are calculated as requests per seconds per client.

Note: If the rate limit has been reached on the management subsystem, the client will get an HTTP error: **429 Too Many Requests**.

Procedure

- Set a rate limit:
 - Add the following entries to the extra-values-file:

```
spec:
  template:
    - name: juhu
      containers:
        - name: juhu
          env:
            - name: RATE_LIMIT_PER_CLIENT
              value: "10"
            - name: LIMIT_REQUEST_OPTION
              value: "burst=10 nodelay"
```

In this example, the first option sets the rate to 10 requests per second (10r/s). The second option allows 10 requests boosted without delay within < 1 second. You can customize as needed.

The `rateLimitPerClient` property sets `rate`, and `limitRequestOption` sets `[burst=number] [nodelay | delay=number]` in the following nginx configuration:

```
limit_req_zone key zone=name:size rate=rate;
limit_req zone=name [burst=number] [nodelay | delay=number];
```

Note that `zone` has been pre-defined and can't be configured. For more details, see the nginx doc http://nginx.org/en/docs/http/nginx_limit_req_module.html.

- Save the extra-values file with a filename of your choice, but it must have a `.yaml` or `.yml` extension.
- Use `apicup` to update the VMware `extra-values-file` attribute to point to the new or updated management subsystem extra-values file:

```
apicup subsys set <mgmt_subsystem_name> extra-values-file <extra_values_file_path_and_name>
```

Where:

- `<mgmt_subsystem_name>` is the name of the management subsystem that you are configuring.
- `<extra_values_file_path_and_name>` is the fully qualified path and name of your extra-values file.

For example:

```
apicup subsys set ds10-management extra-values-file /workspace/v10012/my_extra_values.yaml
```

- Verify that the configuration settings are valid in the `apiconnect-up-v10.yaml` by running the following command:

```
apicup subsys get <mgmt_subsystem_name> --validate
```

The output lists each setting and adds a check mark after the value once the value is validated. If the setting lacks a check mark and indicates an invalid value, reconfigure the setting.

- Run `apicup` to update the settings in the deployed Management subsystem.

```
apicup subsys install <management-subsystem>
```

- Enter the following command to connect to the management subsystem using SSH:

```
ssh ip_address -l apicadm
```

You are logging in with the default ID of `apicadm`, which is the API Connect ID that has administrator privileges.

- Select Yes to continue connecting, and once connected:

```
sudo -i
```

- Validate that the juhu pod has restarted by listing the pods:

```
kubectl get pods grep juhu
```

- Check the AGE column to ensure a new juhu pod has started.
- If the older juhu pod is still running, delete it with the following command:

```
kubectl delete pods <old-juhu-pod>
```

- Disable a rate limit:
 - Remove the values from the extra-values-file.
 - Run `apicup` to update the settings in the deployed Management subsystem.

```
apicup subsys install <management-subsystem>
```

3. Enter the following command to connect to the management subsystem using SSH:

```
ssh ip_address -l apicadm
```

You are logging in with the default ID of *apicadm*, which is the API Connect ID that has administrator privileges.

4. Select Yes to continue connecting, and once connected:

```
sudo -i
```

5. Validate that the juhu pod has restarted by listing the pods:

```
kubectl get pods grep juhu
```

6. Check the AGE column to ensure a new juhu pod has started.
7. If the older juhu pod is still running, delete it with the following command:

```
kubectl delete pods <old-juhu-pod>
```

Configuring maximum size of client requests to the Management subsystem on VMware

You can configure the maximum allowed size of the client request body for requests made to the Management subsystem.

About this task

The default maximum size is 8 megabytes. You can increase this value.

Increasing the setting can be useful if you get errors when using the CLI to import a large API. Example error:

```
HTTP/1.1 413 Request Entity Too Large
```

Note:

- These instructions apply only to VMware installations. For Kubernetes installations, see [Configuring maximum size of client requests to the Management subsystem](#).
- These instructions use an extra-values file. This file can be used to define CR overrides. Note that extra-values files for a CR can contain more than one override. If you already use an extra-values file, you can add the new override to it.
- All of the commands must be run in the **apicup** management project directory.

Procedure

1. Create, or edit if one already exists, a management subsystem extra-values file in the **apicup** management project directory, and enter the following configuration details:

```
spec:
  template:
    - name: juhu
      containers:
        - name: juhu
          env:
            - name: CLIENT_MAX_BODY_SIZE
              value: "12m"
```

2. Save the extra-values file with a filename of your choice, but it must have a .yaml or .yml extension.
3. Use **apicup** to update the VMware **extra-values-file** attribute to point to the new or updated management subsystem extra-values file:

```
apicup subsys set <mgmt_subsystem_name> extra-values-file <extra_values_file_path_and_name>
```

Where:

- **<mgmt_subsystem_name>** is the name of the management subsystem that you are configuring.
- **<extra_values_file_path_and_name>** is the fully qualified path and name of your extra-values file.

For example:

```
apicup subsys set ds10-management extra-values-file /workspace/v10011/my_extra_values.yaml
```

4. Verify that the configuration settings are valid in the apiconect-up.yaml by running the following command:

```
apicup subsys get <mgmt_subsystem_name> --validate
```

The output lists each setting and adds a check mark after the value once the value is validated. If the setting lacks a check mark and indicates an invalid value, reconfigure the setting.

5. Update the management VMware with the updated setting.

```
apicup subsys install <mgmt_subsystem_name>
```

6. Monitor the health-check output until the management subsystem is healthy by running the following command:

```
apicup subsys health-check <mgmt_subsystem_name>
```

If one or more of the health criteria are not met, the command stops processing and displays a message with the failure, and exits with a status of **1**. The following output is an example of unhealthy output while the install is running:

```
Error: Cluster not in good health:
ManagementCluster (current ha mode: active) is not ready | State: 15/17 Phase: Pending
ManagementCluster (current ha mode: active) is not ready | State: 15/17 Phase: Pending
```

When all of the health criteria are successfully met, the command displays no output, and exits with a status of 0.

Configuring monetization on VMware

You can optionally configure monetization in API Connect on a VMware virtual machine by enabling the billing microservice.

Before you begin

Important: If the optional billing microservice is enabled, you must increase the memory for the API Manager for VMware component by 0.5 GB.

About this task

API Connect includes a subscription billing microservice that allows API providers to define pricing plans in their API Products, and monetize their API offerings. If a Product contains a pricing plan, API Consumers must enter their payment information into the Developer Portal before they can subscribe to that plan. API Connect supports integration with Stripe Subscription Billing, an independent cloud service that manages monetized product plans, customers, their payment information, and their subscription history, in order to generate monthly invoices and charge customers automatically. With this integration, Stripe serves as both the subscription billing system and the payment processing system.

Note:

- These instructions apply only to VMware installations. For Kubernetes installations, see [Configuring monetization on Kubernetes](#).
- To use Stripe as your credit card processing vendor, you must have port 443 open to HTTPS communication between the Stripe API and your Developer Portal management and the Management cluster servers. See [Firewall requirements on Kubernetes](#) and [Firewall requirements on VMware](#) for more information about this requirement.
- Provider organizations cannot add any billing integrations until the billing microservice is enabled.
- After provider organizations have added billing integrations, the billing microservice must not be disabled.
- All of the commands must be run in the `apicup` management project directory.

To enable or disable the billing microservice, you must configure a management subsystem extra-values file. See the following instructions:

- [Enabling the billing microservice](#)
- [Disabling the billing microservice](#)

Procedure

- **Enabling the billing microservice**

1. Create, or edit if one already exists, a management subsystem extra-values file in the `apicup` management project directory, and enter the following configuration details:

```
spec:
  billing:
    enabled: true
```

2. Save the extra-values file with a filename of your choice, but it must have a `.yaml` or `.yml` extension.
3. Update the VMware `extra-values-file` attribute to point to the new or updated management subsystem extra-values file. Run the following `apicup` command to insert the location of the extra-values file:

```
apicup subsys set <mgmt_subsystem_name> extra-values-file <extra_values_file_path_and_name>
```

Where:

- `<mgmt_subsystem_name>` is the name of the management subsystem that you are configuring.
- `<extra_values_file_path_and_name>` is the fully qualified path and name of your extra-values file.

For example:

```
apicup subsys set ds10-management extra-values-file /workspace/v10011/monetization.yaml
```

4. Verify that the configuration settings are valid in the `apiconnect-up.yaml` by running the following command:

```
apicup subsys get <mgmt_subsystem_name> --validate
```

The output lists each setting and adds a check mark after the value once the value is validated. If the setting lacks a check mark and indicates an invalid value, reconfigure the setting.

5. Update the management VMware with the updated billing value. Install the new setting for the billing microservice by running the following command:

```
apicup subsys install <mgmt_subsystem_name> --debug
```

Including the `--debug` option enables the debug output for the command.

6. Monitor the health-check output until the management subsystem is healthy by running the following command:

```
apicup subsys health-check <mgmt_subsystem_name>
```

If one or more of the health criteria are not met, the command stops processing and displays a message with the failure, and exits with a status of 1. The following output is an example of unhealthy output while the install is running:

```
Error: Cluster not in good health:
ManagementCluster (current ha mode: active) is not ready | State: 15/17 Phase: Pending
ManagementCluster (current ha mode: active) is not ready | State: 15/17 Phase: Pending
```

When all of the health criteria are successfully met, the command displays no output, and exits with a status of 0. The billing microservice is now enabled, and provider organization owners can create billing integration resources for API providers in the API Manager UI. For more information, see [Monetizing your Products](#).

- **Disabling the billing microservice**

Important: After provider organizations have added billing integrations, the billing microservice must not be disabled.

1. Edit the management subsystem extra-values file in the `apicup` management project directory, and change the `enabled` configuration option to `false`:

```
spec:
  billing:
    enabled: false
```

2. Save the changed extra-values file.
3. Verify that the configuration settings are valid in the `apiconnect-up.yaml` by running the following command:

```
apicup subsys get <mgmt_subsystem_name> --validate
```

The output lists each setting and adds a check mark after the value once the value is validated. If the setting lacks a check mark and indicates an invalid value, reconfigure the setting.

4. Update the management VMware with the altered billing value. Install the updated setting for the billing microservice by running the following command:

```
apicup subsys install <mgmt_subsystem_name> --debug
```

Including the `--debug` option enables the debug output for the command.

5. Monitor the health-check output until the management subsystem is healthy by running the following command:

```
apicup subsys health-check <mgmt_subsystem_name>
```

If one or more of the health criteria are not met, the command stops processing and displays a message with the failure, and exits with a status of 1. The following output is an example of unhealthy output while the install is running:

```
Error: Cluster not in good health:
ManagementCluster (current ha mode: active) is not ready | State: 15/17 Phase: Pending
ManagementCluster (current ha mode: active) is not ready | State: 15/17 Phase: Pending
```

When all of the health criteria are successfully met, the command displays no output, and exits with a status of 0. The billing microservice is now disabled.

Related information

- [Monetizing your Products](#)
- [stripe.com](#)

Configuring use of an external NTP server

You can optionally configure an external NTP server for use by API Connect when deploying on a VMware virtual machine.

About this task

Secure communication between API Connect subsystems relies on the system time being in sync on all hosts. For example, time stamps are checked to ensure that certificates are valid. When API Connect is deployed behind a firewall that blocks access to the internet, the API Connect subsystems cannot by default access a Network Time Protocol (NTP) server.

You can use an additional cloud-init file to manually specify an NTP server for use by the subsystems. Complete the following steps.

Procedure

1. Create the cloud-init file extra values file, and enter the configuration details that you want to overwrite. For example:

```
ntp:
  enabled: true
  ntp_client: systemd-timesyncd
  servers:
    - time.google.com
```

2. Use `apicup` to specify the cloud-init file.

Syntax:

```
apicup subsys set <subsys> additional-cloud-init-file <path-to-cloud-init-file>
```

Example:

```
apicup subsys set mgmt additional-cloud-init-file myCloudInitFile.yaml
```

3. Install the subsystem. Note that the output directory must be empty:

```
apicup subsys install mgmt --out mgmtplan-out
```

4. Deploy the VMware image (.ova) with the ISO file that is generated. To review the deployment steps, see [Deploying the Management virtual appliance](#).
5. Verify that the correct NTP server is being used:

```
journalctl -u systemd-timesyncd
```

Example output for the NTP server that was set in Step 1:

```
Nov 05 21:09:24 h-apicdev-4 systemd[1]: Starting Network Time Synchronization...
Nov 05 21:09:24 h-apicdev-4 systemd[1]: Started Network Time Synchronization.
Nov 05 21:09:24 h-apicdev-4 systemd-timesyncd[1697]: Synchronized to time server 216.239.35.8:123 (time.google.com).
```

Overriding resources for Postgres components on VMware

In 10.0.4.0 and greater, you can optionally override the CPU and Memory resources used by postgres database components.

About this task

There are three postgres components:

- `postgres`
- `pgBouncer`
- `pgBackRest`

The default resources used in `postgres`, `pgBouncer` and `pgBackRest` components are:

Table 1. Postgres component default resources

Resource	Postgres DB	pgBouncer	pgBackRest
Memory Request	512Mi	24Mi	48Mi
Memory Limit	8Gi	128Mi	128Mi
CPU Request	600m	80m	80m
CPU Limit	1500m	200m	200m

Procedure

To override resources, complete the following procedure

1. Create an extra-values yaml file, or add values to an existing extra-values file.
 - Ensure that you specify the name of the container for the resource you want to override. Names:
 - `postgres`
 - `pgBouncer`
 - `pgBackRest`
 - You can override multiple postgres components in parallel. Adding one container section under `mgmt-pgcluster` template.
 - You can override just a single value, such as CPU limit.

Examples:

- Example yaml, to replace all `postgres` resources:

```
spec:
  template:
    - name: mgmt-pgcluster
      containers:
        - name: postgres
          resources:
            requests:
              cpu: 1500m
              memory: 8Gi
            limits:
              cpu: 600m
              memory: 512Mi
```

- Example yaml, to replace all `pgBouncer` resources:

```
spec:
  template:
    - name: mgmt-pgcluster
      containers:
        - name: pgBouncer
          resources:
            requests:
              cpu: 200m
              memory: 128Mi
            limits:
              cpu: 300m
              memory: 256Mi
```

- Example yaml, to replace CPU limit of `postgres` database:

```
spec:
  template:
    - name: mgmt-pgcluster
      containers:
        - name: postgres
          resources:
            limits:
              cpu: 1000m
```

- Example yaml, to replace CPU limits of `postgres` database and `pgBouncer`:


```
spec:
  template:
    - name: mgmt-pgcluster
      containers:
        - name: postgres
          resources:
            limits:
              cpu: 1000m
        - name: pgBouncer
          resources:
            limits:
              cpu: 1000m
```

2. Set the extra values file:

```
apicup subsys set <SUBSYSTEM_NAME> extra-values-file=<PATH_TO_EXTRA_VALUES_FILE>
```

3. Apply the overrides:

```
apicup subsys install <SUBSYSTEM_NAME>
```

4. Check health of cluster:

```
apicup subsys health-check <SUBSYSTEM_NAME>
```

Results

The postgres components will restart with updated values.

Enabling UI feature flags on VMware

Use a template override to update the installation CR with settings that control UI features.

About this task

You can add a template override to the Management subsystem CR and use it to control whether specific features are available in the UI. Some features can be optionally disabled by users (by disabling them in the browser) provided you have enabled the feature for the cluster as a whole. Table 1 describes the available feature flags.

Table 1. Flags and settings for controlling UI features

Feature ID	Default Value	Partial Override For	Description
ffAdvancedPublish	true		Enable advanced publish option (deprecated)
allOmniSearch	true		Enables OmniSearch for all lists. This flag is used in conjunction with other flags.
draftsOmniSearch	true	allOmniSearch	Enable OmniSearch for drafts list
catalogProductsOmniSearch	true	allOmniSearch	Enable OmniSearch for catalog
orgsOmniSearch	true	allOmniSearch	Enable OmniSearch for org lists
appsOmniSearch	true	allOmniSearch	Enable OmniSearch for apps
subscriptionsOmniSearch	true	allOmniSearch	Enable OmniSearch for subscriptions, if disabled, subscription lists will be removed from catalog list
requestedApprovalsOmniSearch	true	allOmniSearch	Enable OmniSearch for requested approvals
approvalTasksOmniSearch	true	allOmniSearch	Enable OmniSearch for approvals
productApisOmniSearch	true	allOmniSearch	Enable OmniSearch for apis within products
membersOmniSearch	true	allOmniSearch	Enable OmniSearch for member lists

Procedure

1. Add the template override to the `management-extra-values.yaml` file.
2. In the file, append the feature-flag definition to the end of the `spec:` section, making sure to adhere to the spacing used in the file. The following example enables the OmniSearch feature on all possible UI pages.

```
spec:
  template:
    - name: ui
      containers:
        - name: ui
          env:
            - name: APIC_UI_FEATURES
              value: allOmniSearch=always
```

`APIC_UI_FEATURES` is a URL-encoded string. Each parameter is formed as `featureId=$featureValue`, and you can list multiple parameters with the `&` delimiter (`featureId=$featureValue&featureId=$featureValue`).

Use the `featureId` values from Table 1. For each feature, specify one of the following `featureValue` settings:

- `true` - On by default, but can be overridden by the user, for their browser alone
- `false` - Off by default, but can be overridden by the user, for their browser alone
- `never` - Off regardless of browser setting, disabled in feature flag page
- `always` - On regardless of browser setting, disabled in feature flag page

3. Save the file.

4. Run the following command to update the deployment:

```
apicup subsys set <management_subsystem> extra-values-file <path_to_management-extra-values.yaml_file>
```

The cluster is updated to enable or disable features as specified.

Backing up and restoring on VMware

Backup and restore API Connect on VMware.

Backups are for recovery of the API Connect subsystems in the same environment that the backups were taken. If you want to move your API Connect deployment to a different environment and change the form factor or modify your API Connect endpoints, then see [Migrating from v10 to v10 on a different form factor](#).

Note: For information about how to maintain a two data center deployment, including backup and restore considerations, see [Maintaining a two data center deployment](#).

- [Backing up and restoring the Management subsystem](#)
You can back up and restore your Management subsystem in your VMware environment.
- [Backing up and restoring the Developer Portal](#)
You can backup and restore a Developer Portal in a VMware environment.
- [Backing up and restoring the Analytics database on VMware](#)
The IBM® API Connect Analytics database can be backed up and restored from an S3 repository. S3 compatible object storage is required; for example, IBM Cloud Object Storage.

Backing up and restoring the Management subsystem

You can back up and restore your Management subsystem in your VMware environment.

About this task

- You use the `apicup` utility to configure backup settings for the management subsystem.
- Database backups can be used to restore the database for disaster recovery.
- Backups can be scheduled or generated manually (on-demand). You can then list what backups are available and then use one to restore your Management subsystem if required.
- Postgres replica pods depend on a successful completed backup. If backup configuration is not correct, replica pods won't come up.
- Restoring a backup restores the registration credentials (client_ID, client_secret) that were in use at the time that the selected backup was created. For information on the registration credentials, see [Changing the registration client id and client_secret for applications](#).

Note: You must backup both the Management and Portal subsystems at the same time, to ensure synchronicity across the services.

Types of backups supported:

- Cloud-based S3 storage
 - IBM Cloud Object Storage
 - AWS S3
 - **v10.0.2.0 or later:** Any custom S3, such as minIO
- SFTP
- Local Backups
- [Configuring backup settings during initial installation of the management subsystem](#)
You can use the `apicup` utility to configure backup settings during initial installation of the management subsystem
- [Verify configuration for s3 backup](#)
Verify that the configuration for s3 backups succeeded.
- [Reconfiguring backup settings for the management subsystem](#)
You can use the `apicup` utility to add or reconfigure backup settings for the management subsystem, after completion of initial installation.
- [Backing up the management subsystem](#)
Use `apicup` to create a manual backup of the management subsystem
- [Restoring the management subsystem](#)
Use `apicup` to list backups and restore a selected backup of the management subsystem
- [Troubleshooting management subsystem backups on VMware](#)
You can troubleshoot failures of the management subsystem backups.

Configuring backup settings during initial installation of the management subsystem

You can use the `apicup` utility to configure backup settings during initial installation of the management subsystem

Before you begin

Review [Backing up and restoring the Management subsystem](#).

Note: You must backup both the Management and Portal subsystems at the same time, to ensure synchronicity across the services.

About this task

- During an initial installation, use these steps to configure the backup settings prior to creating the initial management ISO.
- You can configure scheduled backups, or generate manual (on-demand) backups.
- Changing backup settings results in database downtime, unless `database-backup-protocol=sftp`.
- Backup to S3 providers of type `custom` is supported in v10.0.2.0 or greater. For `custom` S3, provide values for `database-backup-certs` and `database-backup-s3-uri-style`.
- OpenSSH key authentication for SFTP is supported.

Important: For S3 backups of the management subsystem, do not use retention features provided by Cloud-based S3 storage providers. Use of such features can result in periodic deletion of archived backups, which can cause backup corruptions which can cause database restores to fail.

Procedure

1. Use `apicup` to specify backup settings:

- For most settings, use `apicup subsys:`

```
apicup subsys set <subsystem_name> <setting>=<value>
```

- When configuring ssh-key authentication for SFTP, use `apicup certs`

```
apicup certs set <subsystem_name> database-backup-auth-ssh-key <ssh-key-file-path>
```

Table 1. Configuration settings for backup of management subsystem

Parameter	Description
<code>database-backup-protocol</code>	<p>The type of the backup. Supported types:</p> <ul style="list-style-type: none"> • <code>objstore</code> <pre>apicup subsys set mgmt database-backup-protocol=objstore</pre> <ul style="list-style-type: none"> • <code>sftp</code> <pre>apicup subsys set mgmt database-backup-protocol=sftp</pre> <p>The default value is <code>sftp</code>.</p> <ul style="list-style-type: none"> • <code>local</code> - Local storage backups <pre>apicup subsys set mgmt database-backup-protocol=local</pre> <p>Note:</p> <ul style="list-style-type: none"> • The public certificate on the S3 storage provider must be signed by a known certificate authority that is trusted by API Connect. Use of an untrusted authority can cause the following error during backup upload: <code>x509: certificate signed by unknown authority</code>.
<code>database-backup-auth-pass</code>	<p>The password for the server specified in <code>database-backup-host</code>. If using object store, this would be the S3 Secret Access Key parameter. The password will be stored in Base64 encoded format. For example:</p> <pre>apicup subsys set mgmt database-backup-auth-pass '<password>'</pre> <p>Note that you cannot use the <code>`</code> sign to assign the password to <code>database-backup-auth-pass</code>.</p>
<code>database-backup-auth-user</code>	<p>The username for the server specified in <code>database-backup-host</code>. If using object store, this would be the S3 Secret Key ID.</p> <pre>apicup subsys set mgmt database-backup-auth-user <user_name></pre>
<code>database-backup-auth-ssh-key</code>	<p>Optional: OpenSSH private key for DB backups using SFTP only. Changing this value will restart the DB. Important:</p> <ul style="list-style-type: none"> • Use <code>apicup certs</code> to set this value. Do not use <code>apicup subsys</code>. • Only OpenSSH keys are supported.¹ <pre>apicup certs set mgmt database-backup-auth-ssh-key <ssh private key filepath></pre>
<code>database-backup-s3provider</code>	<p>When using <code>objstore</code> type backup, specify the name of the S3 provider to use. Specify one of the supported values:</p> <ul style="list-style-type: none"> • <code>ibm</code> • <code>aws</code> • <code>custom - v10.0.2.0 or later</code> <p>Not required for local backups or SFTP backups.</p> <pre>apicup subsys set mgmt database-backup-s3provider=ibm</pre>
<code>database-backup-host</code>	<p>The backups host:</p> <ul style="list-style-type: none"> • For <code>objstore</code> type backup, specify S3 endpoint with the corresponding S3 region in the format <code><S3endpoint>/<S3region></code> <pre>apicup subsys set mgmt database-backup-host=s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard</pre> <ul style="list-style-type: none"> • For <code>sftp</code> type, the SFTP server hostname <pre>apicup subsys set mgmt database-backup-host=my.sftp.backup.domain.example.com</pre> <ul style="list-style-type: none"> • Not required for local storage backups

Parameter	Description
<code>database-backup-path</code>	<p>The path to the location of the backup:</p> <ul style="list-style-type: none"> For <code>objstore</code> backups, the name of your S3 bucket to store backup data. For example, <code>bucket-name/folder</code>. The use of subdirectories in the bucket name is not supported. <pre>apicup subsys set mgmt database-backup-path=apic-backup</pre> <p>Note: When your deployment includes a management subsystem in two different clusters (with active databases), the two management subsystems cannot use the same s3 bucket name in the database backup configurations. Each management subsystem must use a unique s3 bucket name.</p> <p>Ensure that <code>bucket-name/folder</code> is empty. If the folder was previously used for backups, the folder will not be empty, and stanza create might encounter an error.</p> <ul style="list-style-type: none"> Explanation: Once a folder is used, <code>archive.info</code> and <code>backup.info</code> files are created. During stanza creation, the process compares the database version and database system id between the two info files and the current Postgres database cluster. Stanza creation fails if there is a mismatch. To prevent this possibility, remove all files from the folder before configuring. For <code>sftp</code> type, the full absolute path of the folder on the SFTP server, beginning with <code>/</code>. <pre>apicup subsys set mgmt database-backup-path=<folder_name></pre> <ul style="list-style-type: none"> Not required for local storage backups
<code>database-backup-port</code>	<p>The port for the protocol to connect to the <code>database-backup-host</code>. The backup port is not required for object storage. Default: 22</p> <pre>apicup subsys set mgmt database-backup-port=<port_number></pre>
<code>database-backup-retries</code>	<p>For <code>sftp</code> type backups, the number of times the <code>ibm-apiconnect</code> Operator attempts backups in the event of a failed SFTP backup.</p> <pre>apicup subsys set mgmt database-backup-retries=3</pre> <p>Default value: 0.</p> <p>Not used for <code>objstore</code> or local storage backup.</p>
<code>database-backup-schedule</code>	<p>Cron like schedule for performing automatic backups. The format for the schedule is:</p> <ul style="list-style-type: none"> * * * * * ----- +----- day of week (0 - 6) (Sunday=0) +----- month (1 - 12) +----- day of month (1 - 31) +----- hour (0 - 23) +----- min (0 - 59) <p>The backup schedule defaults to <code>0 0 * * *</code>. This means a backup is run every day at midnight and minute zero.</p> <p>The timezone for backups is UTC.</p> <p>For local backups, the default is 01:00 UTC.</p> <p>When you configure a host, if you do not specify a value for <code>database-backup-schedule</code>, the default backup schedule is automatically set. Note that the default backup schedule is not set, and scheduled backups not enabled, until host configuration is completed.</p> <pre>apicup subsys set mgmt database-backup-schedule="0 0 * * *"</pre>
<code>database-backup-certs</code>	<p>Custom certificate. Used only when <code>s3Provider</code> is set to <code>custom</code>. Most of the custom S3 providers are created based on custom CA certificate.</p> <p>This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be <code>ca.crt</code> and value should be base64 encoded value of CA certificate.</p> <p>The API Connect management subsystem validates the custom S3 certificate against the customer-provided CA certificate.</p> <p>Sample bash script to create the Kubernetes secret:</p> <pre>cat >customs3ca.yaml <<EOF apiVersion: v1 data: ca.crt: \$(base64 <path-to-ca-certificate> tr -d '\n') kind: Secret metadata: name: custom-server-ca type: generic EOF kubectl apply -f customs3ca.yaml -n <namespace></pre> <p>For example:</p> <pre>apicup subsys set mgmt database-backup-certs=my-custom-s3-ca-cert</pre>
<code>database-backup-s3-uri-style</code>	<p>Valid values: <code>host</code> and <code>path</code>. Default value: <code>host</code></p> <p>Only allowed if <code>s3Provider</code> is set to <code>custom</code>.</p> <p>Some custom S3 providers require URI style to be set to <code>path</code>. For example, minIO supports both <code>host</code> and <code>path</code> style setup. You can create a minIO S3 server to only accept <code>path</code> style client communications.</p> <p>Important: Contact your custom S3 administrator before configuring this field. If not properly configured, upstream custom S3 can reject connections from the client, such as API Connect management subsystem.</p> <pre>apicup subsys set mgmt database-backup-s3-uri-style='<host path>'</pre>

Parameter	Description
<code>database-backup-repo-retention-full</code>	<p>The number of full S3 or local backups to retain. Not supported for SFTP.</p> <p>When the next full backup successfully completes, and the specified number of retained backups is reached, the oldest full backup is deleted. All incremental backups and archives associated with the oldest full backup also expire. Incremental backups are not counted for this setting.</p> <p>Applies to both manual backups and scheduled backups.</p> <p>Minimum value: 1</p> <p>Maximum value: 9999999</p> <p>Default: none (S3 backups), 14 (local backups).</p>

2. Verify that your settings are valid:

- To view all subsystem settings except `database-backup-auth-ssh-key`:

```
apicup subsys get <subsystem_name> --validate
```

- To view `database-backup-auth-ssh-key`:

```
apicup certs get <subsystem_name> database-backup-auth-ssh-key --type key
```

3. Return to the instructions for installing the management subsystem: [Deploying the Management virtual appliance](#).

Note that the configuration settings for backups will be installed when you run `apicup subsys install <subsystem_name>` after all configuration settings are complete.

4. **S3 backups only:** After you complete the deployment of the Management subsystem in the previous step, continue to [Verify configuration for s3 backup](#).

¹ PuTTY style keys can be converted to OpenSSH by using the PuTTY Key Generator (PuTTYgen) application; see <https://www.puttygen.com/>.

Verify configuration for s3 backup

Verify that the configuration for s3 backups succeeded.

About this task

When you configure Management subsystem backups, the operator runs a `stanza-create` job. This job creates the stanza (Postgres cluster configuration) on the upstream S3 server, which is used for backup and archive procedures. The job also brings up the necessary pod.

Procedure

- SSH into the management appliance.
- Check the status of the `stanza-create` job:

```
kubectl get jobs -n <namespace> | grep stanza
```

- On success:

- The `stanza-create` job status is 1/1:

```
kubectl get jobs | grep stanza
m1-b722e361-postgres-stanza-create      1/1      5s      127m
```

- The `stanza-create` job shows `status: "True"` and `type: Complete`:

```
kubectl get job m1-b722e361-postgres-stanza-create -o yaml
```

```
status:
  completionTime: "2021-04-13T18:38:50Z"
  conditions:
  - lastProbeTime: "2021-04-13T18:38:50Z"
    lastTransitionTime: "2021-04-13T18:38:50Z"
    status: "True"
    type: Complete
  startTime: "2021-04-13T18:38:45Z"
  succeeded: 1
```

- The `stanza-create` pod is in `Completed` state:

```
kubectl get pods | grep stanza
m1-b722e361-postgres-stanza-create-q4fvp 0/1      Completed 0      127m
```

- Exec into the pod:

```
kubectl exec -it <backrest-shared-repo-pod> -- bash
```

- The `pgbackrest` command returns the S3 contents in valid JSON:

```
pgbackrest info --output json --repol-type s3
```

```
"backup":{"held":false},"message":"ok"}} [pgbackrest@m1-b722e361-postgres-backrest-shared-repo-69cdfdfdd5-rs882 /
```

- On failure:

- `stanza-create` pods fail, and job status is 0/1:

```
kubectl get jobs | grep stanza
m1-3bfe12ac-postgres-stanza-create 0/1 32m 32m
```

- Listing all pods brought up by `stanza-create` job shows multiple pods because a job tries `backoffLimit` times to complete the job. By default, there will be 7 pods in error state:

```
kubectl get pods | grep stanza
m1-3bfe12ac-postgres-stanza-create-726z4 0/1 Error 0 11m
m1-3bfe12ac-postgres-stanza-create-9vq4n 0/1 Error 0 22m
m1-3bfe12ac-postgres-stanza-create-gbtgb 0/1 Error 0 30m
m1-3bfe12ac-postgres-stanza-create-g15g5 0/1 Error 0 17m
m1-3bfe12ac-postgres-stanza-create-q8qhw 0/1 Error 0 26m
m1-3bfe12ac-postgres-stanza-create-t84zs 0/1 Error 0 8m54s
m1-3bfe12ac-postgres-stanza-create-z8tfs 0/1 Error 0 12m
```

Note: In some cases, the pods are cleaned by Kubernetes and the logs of these pods can be lost.

- Pod log errors show a reason for the failure. For example, when a hostname is not valid:

```
kubectl logs m1-3bfe12ac-postgres-stanza-create-gbtgb

time="2021-04-13T20:01:19Z" level=info msg="pgo-backrest starts"
time="2021-04-13T20:01:19Z" level=info msg="debug flag set to false"
time="2021-04-13T20:01:19Z" level=info msg="backrest stanza-create command requested"
time="2021-04-13T20:01:19Z" level=info msg="s3 flag enabled for backrest command"
time="2021-04-13T20:01:19Z" level=info msg="command to execute is [pgbackrest stanza-create --db-host=192.1.2.3 --db-path=/pgdata/m1-3bfe12ac-postgres --repo1-type=s3]"
time="2021-04-13T20:01:19Z" level=info msg="command is pgbackrest stanza-create --db-host=192.1.2.3 --db-path=/pgdata/m1-3bfe12ac-postgres --repo1-type=s3 "
time="2021-04-13T20:05:21Z" level=error msg="command terminated with exit code 49"
time="2021-04-13T20:05:21Z" level=info msg="output=[]"
```

- Job status is set to `type: Failed` and `status: "True"`, with `reason: BackoffLimitExceeded`:

```
kubectl get job m1-3bfe12ac-postgres-stanza-create -o yaml

status:
  conditions:
  - lastProbeTime: "2021-04-13T20:28:03Z"
    lastTransitionTime: "2021-04-13T20:28:03Z"
    message: Job has reached the specified backoff limit
    reason: BackoffLimitExceeded
    status: "True"
    type: Failed
    failed: 7
  startTime: "2021-04-13T20:01:18Z"
```

- If pod logs are lost, you can execute a command inside the `backrest-shared-repo` pod:
 - Obtain the `backrest-shared-repo` pod name:

```
kubectl get pods -n <namespace> | grep backrest-shared-repo
```

For example:

```
kubectl get pods | grep backrest-shared-repo
m1-eb8edc18-postgres-backrest-shared-repo-98dd46cc6-tw195 1/1 Running
```

- Exec into the pod:

```
kubectl exec -it <backrest-shared-repo-pod> -- bash
```

- Run:

```
pgbackrest info --output json --repo1-type s3
```

For example, for invalid hostname:

```
ERROR: [049]: unable to get address for 'test1-v10-backup.s3.exampleaws.com': [-2] Name or service not knownCopy
```

The exact `ERROR`: will vary depending on the misconfiguration. Common errors:

```
Invalid access key
Invalid access key secret
Invalid bucket region
Invalid bucket or folder path
```

- To fix the incorrect settings, see [Reconfiguring backup settings for the management subsystem](#).

Reconfiguring backup settings for the management subsystem

You can use the `apicup` utility to add or reconfigure backup settings for the management subsystem, after completion of initial installation.

About this task

Warning: If you use the S3 backup protocol, changes to the backup settings result in a short period of database downtime.

- After initial installation, use these steps to add or reconfigure the backup settings.
- When you add or reconfigure backups after completion of the initial installation, the steps are the same as configuring during the initial installation, with the addition of using **apicup** to install (activate) the new settings immediately after adjusting the settings. During initial configuration, you typically wait to use **apicup** to activate settings until all other subsystem settings (unrelated to backup) are complete.
- You can add or reconfigure scheduled backups, or generate manual (on-demand) backups.

Note: You must backup both the Management and Portal subsystems at the same time, to ensure synchronicity across the services.

Procedure

1. Use **apicup** to specify backup settings:

```
apicup subsys set <subsystem_name> <setting>=<value>
```

Table 1. Configuration settings for backup of management subsystem

Parameter	Description
<code>database-backup-protocol</code>	<p>The type of the backup. Supported types:</p> <ul style="list-style-type: none"> • objstore <code>apicup subsys set mgmt database-backup-protocol=objstore</code> • sftp <code>apicup subsys set mgmt database-backup-protocol=sftp</code> The default value is sftp. • local - Local storage backups <code>apicup subsys set mgmt database-backup-protocol=local</code> <p>Note:</p> <ul style="list-style-type: none"> • The public certificate on the S3 storage provider must be signed by a known certificate authority that is trusted by API Connect. Use of an untrusted authority can cause the following error during backup upload: x509: certificate signed by unknown authority.
<code>database-backup-auth-pass</code>	<p>The password for the server specified in <code>database-backup-host</code>. If using object store, this would be the S3 Secret Access Key parameter. The password will be stored in Base64 encoded format. For example: <code>apicup subsys set mgmt database-backup-auth-pass '<password>'</code></p> <p>Note that you cannot use the '=' sign to assign the password to <code>database-backup-auth-pass</code>.</p>
<code>database-backup-auth-user</code>	<p>The username for the server specified in <code>database-backup-host</code>. If using object store, this would be the S3 Secret Key ID. <code>apicup subsys set mgmt database-backup-auth-user <user_name></code></p>
<code>database-backup-auth-ssh-key</code>	<p>Version 10.0.3.0 or later: Optional: OpenSSH private key for DB backups using SFTP only. Changing this value will restart the DB. Important:</p> <ul style="list-style-type: none"> • Use <code>apicup certs</code> to set this value. Do not use <code>apicup subsys</code>. • Only OpenSSH keys are supported.¹ <p><code>apicup certs set mgmt database-backup-auth-ssh-key <ssh private key filepath></code></p>
<code>database-backup-s3provider</code>	<p>When using objstore type backup, specify the name of the S3 provider to use. Specify one of the supported values:</p> <ul style="list-style-type: none"> • ibm • aws • custom <p>Not required for local backups or SFTP backups. <code>apicup subsys set mgmt database-backup-s3provider=ibm</code></p>
<code>database-backup-host</code>	<p>The backups host:</p> <ul style="list-style-type: none"> • For objstore type backup, specify S3 endpoint with the corresponding S3 region in the format <code><S3endpoint>/<S3region></code> <code>apicup subsys set mgmt database-backup-host=s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard</code> • For sftp type, the SFTP server hostname <code>apicup subsys set mgmt database-backup-host=my.sftp.backup.domain.example.com</code> • Not required for local storage backups

Parameter	Description
<code>database-backup-path</code>	<p>The path to the location of the backup:</p> <ul style="list-style-type: none"> For <code>objstore</code> backups, the name of your S3 bucket to store backup data. For example, <code>bucket-name/folder</code>. The use of subdirectories in the bucket name is not supported. <pre>apicup subsys set mgmt database-backup-path=apic-backup</pre> <p>Note: When your deployment includes a management subsystem in two different clusters (with active databases), the two management subsystems cannot use the same s3 bucket name in the database backup configurations. Each management subsystem must use a unique s3 bucket name.</p> <p>Ensure that <code>bucket-name/folder</code> is empty. If the folder was previously used for backups, the folder will not be empty, and stanza create might encounter an error.</p> <ul style="list-style-type: none"> Explanation: Once a folder is used, <code>archive.info</code> and <code>backup.info</code> files are created. During stanza creation, the process compares the database version and database system id between the two info files and the current Postgres database cluster. Stanza creation fails if there is a mismatch. To prevent this possibility, remove all files from the folder before configuring. <ul style="list-style-type: none"> For <code>sftp</code> type, the folder name on the SFTP server <pre>apicup subsys set mgmt database-backup-path=<folder_name></pre> <ul style="list-style-type: none"> Not required for local storage backups
<code>database-backup-port</code>	<p>The port for the protocol to connect to the <code>database-backup-host</code>. The backup port is not required for object storage. Default: 22</p> <pre>apicup subsys set mgmt database-backup-port=<port_number></pre>
<code>database-backup-retries</code>	<p>For <code>sftp</code> type backups, the number of times the <code>ibm-apiconnect</code> Operator attempts backups in the event of a failed SFTP backup.</p> <pre>apicup subsys set mgmt database-backup-retries=3</pre> <p>Default value: 0.</p> <p>Not used for <code>objstore</code> or local storage backup.</p>
<code>database-backup-schedule</code>	<p>Cron like schedule for performing automatic backups. The format for the schedule is:</p> <ul style="list-style-type: none"> * * * * * ----- +----- day of week (0 - 6) (Sunday=0) +----- month (1 - 12) +----- day of month (1 - 31) +----- hour (0 - 23) +----- min (0 - 59) <p>The backup schedule defaults to <code>0 0 * * *</code>. This means a backup is run every day at midnight and minute zero.</p> <p>The timezone for backups is UTC.</p> <p>When you configure a host, if you do not specify a value for <code>database-backup-schedule</code>, the default backup schedule is automatically set. Note that the default backup schedule is not set, and scheduled backups not enabled, until host configuration is completed.</p> <pre>apicup subsys set mgmt database-backup-schedule="0 0 * * *"</pre>
<code>backup-certs</code>	<p>Optional. Backup certificates to use for TLS communication between API Connect and the backup server. Supported for custom s3 backups only. Use this setting when you have set <code>database-backup-s3provider=custom</code></p> <p>For example:</p> <pre>apicup subsys set mgmt backup-certs=<certificate_name></pre>
<code>database-backup-s3-uri-style</code>	<p>Optional: Specifies whether the URI is in the form of <code>host</code> or <code>path</code>. Used only with custom s3providers. Use this setting when you have set <code>database-backup-s3provider=custom</code>.</p> <pre>apicup subsys set mgmt database-backup-s3-uri-style='<host path>'</pre> <p>Supported values:</p> <ul style="list-style-type: none"> <code>host</code> <code>path</code>
<code>database-backup-repo-retention-full</code>	<p>The number of full S3 or local backups to retain. Not supported for SFTP.</p> <p>When the next full backup successfully completes, and the specified number of retained backups is reached, the oldest full backup is deleted. All incremental backups and archives associated with the oldest full backup also expire. Incremental backups are not counted for this setting.</p> <p>Applies to both manual backups and scheduled backups.</p> <p>Minimum value: 1</p> <p>Maximum value: 9999999</p> <p>Default: none (S3 backups), 14 (local backups).</p>

Note: To view all subsystem settings for the management subsystem:

```
apicup subsys get <subsystem_name> --validate
```

2. Apply the settings:

```
apicup subsys install <subsystem_name>
```

Note: The `apicup subsys install` command automatically runs `apicup health-check` prior to installing the new configuration. An error is displayed if a problem is found that will prevent successful reconfiguration.

In some scenarios, after you fix the problem, the health check might continue to fail. In this case, you can use the `--skip-health-check` flag to suppress the automatic health check:


```
apicup subsys install <subsystem_name> --skip-health-check
```

3. Ensure that the changes were applied successfully:

```
apicup subsys health-check <subsystem_name>
```

¹ PuTTY style keys can be converted to OpenSSH by using the PuTTY Key Generator (PuTTYgen) application; see <https://www.puttygen.com/>.

Backing up the management subsystem

Use `apicup` to create a manual backup of the management subsystem

Before you begin

You must have configured backups for the subsystem. See [Configuring backup settings during initial installation of the management subsystem](#)

About this task

Use this task to create an on-demand, or manual, backup of the management system.

To set up scheduled backups, see the `schedule` option in [Configuring backup settings during initial installation of the management subsystem](#).

Procedure

1. Ensure that there are no Management subsystem backups in Running state before creating a new Management backup.
2. Ensure your system is healthy:

```
apicup subsys health-check [subsystem_name]
```

When successful, the command returns silently with an exit code of 0.

3. Make sure that the management cluster status is Running and the READY condition displays the same value before and after the "/". SSH into the appliance and run `kubect1`:

```
kubect1 get mgmt
NAME   READY   STATUS    VERSION   RECONCILED VERSION   AGE
mgmt   16/16   Running   10.0.2.0   10.0.2.0               19h
```

Note: Be sure **READY** status is 16/16. If you have just configured or reconfigured the backup settings, there is a period of time when the postgres pods are taken offline and brought back up with the new configuration. During this time, the `kubect1 get mgmt` command will show 11/11 instead of 16/16. If you start a backup before the Management cluster is on 16/16, the backup attempt might stall indefinitely.

4. Create a backup:

```
apicup subsys backup [subsystem_name] [flags]
```

Decide which of the two types of manual backups you want to do:

- **full**: backs up the entire Management subsystem database.

```
apicup subsys backup management --type full
```

- **incr**: backs up any data that has not been backed up since the last **full** or **incr** backup.

```
apicup subsys backup management --type incr
```

To view additional flags, use

`-h`:

```
./apicup subsys backup management -h
Backup management subsystem
Usage:
  backup SUBSYS [flags]Flags:
  --debug           Enable debug logging
  -h, --help       help for backup
  --type string    Type of backup to take ['incr' (backup data starting from last backup), 'full' (complete backup)] (default "full")
  --wait           Wait for the operation to complete or fail.
  --wait-timeout duration Command timeout in seconds. (default 40s)
```

Restoring the management subsystem

Use `apicup` to list backups and restore a selected backup of the management subsystem

Before you begin

- If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. The backups of the Management and Portal must be taken at the same time to ensure that the Portal sites are consistent with Management database.
- When restoring, the Gateway and all deployed subsystems (Management, Analytics, and Developer Portal) must be at the same version level.
- Restoring the Management Service requires database downtime and is a destructive process that deletes current data and copies backup data. During the restoration process, external traffic must be stopped.

- In a Disaster Recovery scenario, do not log in to the administration UI or attempt to configure or change any settings prior to restoring the backup. Restore the backup immediately after installing the subsystem.
- To restore the management database, you must use the original project directory that was created with **apicup** during the initial product installation. You cannot restore the database without the initial project directory because it contains pertinent information about the cluster. The endpoints and certificates cannot change; the same endpoints and certificates will be used in the restored system. Successful restoration depends on use of a *single apicup* project for all subsystems, even those in a different cluster. Multiple projects will result in multiple certificate chains which will not match.
- Endpoints for the components cannot change between deployments. However, the endpoints for the VMware hosts can be modified for the new deployment.
- Map the DNS entries from the source cluster to the corresponding IP addresses on the target cluster. Record the DNS entries for each endpoint before starting the restore.
- When restoring the management database, the endpoints (on the new cluster which is the target for the restoration) have to be the same as those on the old cluster (the source of the backup).

About this task

Restoring a backup restores the registration credentials (client_ID, client_secret) that were in use at the time that the selected backup was created. For information on the registration credentials, see [Changing the registration client_id and client_secret for applications](#).

Note: There is a known issue where sometimes a restore from the Management subsystem's SFTP backup succeeds but the data is not restored. If this happens, run the restore again.

Procedure

1. To list the available backups:

```
apicup subsys list-backups [subsystem_name] [flags]
```

Example:

```
# apicup subsys list-backups mgmt
NAME          STATUS  ID          CLUSTER  TYPE  CR TYPE  AGE
management-3aa3bebf  Ready  20200929-152150F  management  full  record  10h
management-3c4ca8df  Ready  20200929-115520F  management  full  record  20h
management-5tbxj    Ready  20200929-224349F  management  full  create  17h
management-f10af337  Ready  20200925-133703F  management  full  record  5d2h
management-jtlcd    Ready  20200929-224349F  management  full  create  25h
management-zxjtz    Ready  20200929-224349F  management  full  create  28h
```

Optional flags:

```
./apicup subsys list-backups -h
List backups of the subsystem
Usage:
  apicup subsys list-backups SUBSYS [flags]
Flags:
  -h, --help                help for list-backups
  --timeout duration        Command timeout in seconds. (default 40s)
Global Flags:
  --accept-license          Accept the license for API Connect
  --debug                   Enable debug logging
```

Attention: You can restore the Management subsystem from any backup where the STATUS is Ready; however, you should avoid restoring to the initial system backup. During installation, the Management database is used for an initial system backup before certain database schema jobs are complete. Restoring to this backup will result in an unstable system.

2. To restore a selected backup:

```
apicup subsys restore [subsystem_name] --name <backup_name> [flags]
```

Example, where `management-zxjtz` is the value of `<backup_name>`, chosen from the backup NAMES listed by `apicup subsys list-backups` in Step 1:

```
apicup subsys restore mgmt --name management-zxjtz
```

Optional flags:

```
apicup subsys restore management -h
Restore management subsystem
Usage:
  restore SUBSYS_NAME [flags]
Flags:
  --debug                   Enable debug logging
  -h, --help                help for restore
  --name string             Name of the backup to restore
  --wait                   Wait for the operation to complete or fail.
  --wait-timeout duration  Command timeout in seconds. (default 40s)
```

3. You can list your restores and their statuses.

```
apicup subsys list-restores [subsystem_name] [flags]
```

To view the optional flags, use `-h`:

```
apicup subsys list-restores -h
List restores of the subsystem
Usage:
  apicup subsys list-restores SUBSYS [flags]
Flags:
```

```


-h, --help                help for list-restores
--timeout duration        Command timeout in seconds. (default 40s)

Global Flags:
--accept-license          Accept the license for API Connect
--debug                   Enable debug logging

```

Troubleshooting management subsystem backups on VMware

You can troubleshoot failures of the management subsystem backups.

Note: In the  Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
See:

- [Troubleshooting resiliency issues](#)
Resolve resiliency issues with API Connect management system backups.
- [Troubleshooting stanza-create job for S3 backup configuration](#)
Troubleshoot configuration of s3 backups when the stanza-create job fails.

Troubleshooting resiliency issues

Resolve resiliency issues with API Connect management system backups.

pgbackrest-shared-repo resiliency problems

If the Kubernetes node hosting pgbackrest-shared-repo pod is down and the PVC attached to the pod has strict zone requirements (for example, in AWS or other clouds) or if the storage class is set to `local-storage`, then pgbackrest-shared-repo pod will not get rescheduled to another Kubernetes node.

As a result, there is a single point of failure and the following conditions might occur:

- Backups of the management database fails
- Disk space fills with accumulated Postgres wal (Write-Ahead Logging) files

To avoid this problem, monitor the disk usage: [Monitoring Postgres disk usage on VMware](#).

When the Kubernetes node comes back up, the pod is scheduled and all the processes should resume properly.

Troubleshooting stanza-create job for S3 backup configuration

Troubleshoot configuration of s3 backups when the stanza-create job fails.

When you configure Management subsystem backups for s3 providers, the operator runs a `stanza-create` job. This job creates the stanza (Postgres cluster configuration) on the upstream S3 server, which is used for backup and archive procedures. The job also brings up the necessary pod.

SSH into the appliance VM, and check the status of the `stanza-create` job:

```
kubect1 get jobs -n <namespace> | grep stanza
```

On failure:

- `stanza-create` pods fail, and job status is 0/1:

```
kubect1 get jobs | grep stanza
m1-3bfe12ac-postgres-stanza-create      0/1      32m      32m
```

- Listing all pods brought up by `stanza-create` job shows multiple pods because a job tries `backoffLimit` times to complete the job. By default, there will be 7 pods in error state:

```
kubect1 get pods | grep stanza
m1-3bfe12ac-postgres-stanza-create-726z4      0/1      Error    0      11m
m1-3bfe12ac-postgres-stanza-create-9vq4n      0/1      Error    0      22m
m1-3bfe12ac-postgres-stanza-create-gbtgb      0/1      Error    0      30m
m1-3bfe12ac-postgres-stanza-create-g15g5      0/1      Error    0      17m
m1-3bfe12ac-postgres-stanza-create-q8qhw      0/1      Error    0      26m
m1-3bfe12ac-postgres-stanza-create-t84zs      0/1      Error    0      8m54s
m1-3bfe12ac-postgres-stanza-create-z8tfs      0/1      Error    0      12m
```

Note: In some cases, the pods are cleaned by Kubernetes and the logs of these pods can be lost.

- Pod log errors show a reason for the failure. For example, when a hostname is not valid:

```
kubect1 logs m1-3bfe12ac-postgres-stanza-create-gbtgb

time="2021-04-13T20:01:19Z" level=info msg="pgo-backrest starts"
time="2021-04-13T20:01:19Z" level=info msg="debug flag set to false"
time="2021-04-13T20:01:19Z" level=info msg="backrest stanza-create command
```

```

requested"
time="2021-04-13T20:01:19Z" level=info msg="s3 flag enabled for backrest command"
time="2021-04-13T20:01:19Z" level=info msg="command to execute is [pgbackrest
stanza-create --db-host=192.1.2.3 --db-path=/pgdata/ml-3bfe12ac-postgres --
repol-type=s3]"
time="2021-04-13T20:01:19Z" level=info msg="command is pgbackrest stanza-create --
db-host=192.1.2.3 --db-path=/pgdata/ml-3bfe12ac-postgres --repol-type=s3 "
time="2021-04-13T20:05:21Z" level=error msg="command terminated with exit code 49"
time="2021-04-13T20:05:21Z" level=info msg="output=[]"

```

- Job status is set to `type: Failed` and `status: "True"`, with reason: `BackoffLimitExceeded`:

```

kubect1 get job ml-3bfe12ac-postgres-stanza-create -o yaml

status:
  conditions:
  - lastProbeTime: "2021-04-13T20:28:03Z"
    lastTransitionTime: "2021-04-13T20:28:03Z"
    message: Job has reached the specified backoff limit
    reason: BackoffLimitExceeded
    status: "True"
    type: Failed
    failed: 7
  startTime: "2021-04-13T20:01:18Z"

```

- If pod logs are lost, you can execute a command inside the `backrest-shared-repo` pod:
 1. Obtain the `backrest-shared-repo` pod name:

```
kubect1 get pods -n <namespace> | grep backrest-shared-repo
```

For example:

```
kubect1 get pods | grep backrest-shared-repo
ml-eb8edc18-postgres-backrest-shared-repo-98dd46cc6-tw195 1/1 Running
```

2. Exec into the pod:

```
kubect1 exec -it <backrest-shared-repo-pod> -- bash
```

3. Run:

```
pgbackrest info --output json --repol-type s3
```

For example, for invalid hostname:

```
ERROR: [049]: unable to get address for 'test1-v10-backup.s3.exampleaws.com': [-2] Name or service not knownCopy
```

The exact **ERROR**: will vary depending on the misconfiguration. Common errors:

```

Invalid access key
Invalid access key secret
Invalid bucket region
Invalid bucket or folder path

```

- To fix the incorrect settings, see [Reconfiguring backup settings for the management subsystem](#).

Backing up and restoring the Developer Portal

You can backup and restore a Developer Portal in a VMware environment.

About this task

It is strongly recommended that you configure the backup parameters for your portal service during installation. If you did not do so when you installed API Connect in your runtime environment, you must configure the backups for the Developer Portal before performing an upgrade. These backups can then be used to restore the Developer Portal if required. When your Developer Portal subsystem is running, you can also make on-demand backups by using the command line.

Note:

- You must backup both the Management and Portal subsystems at the same time, to ensure synchronicity across the services.
- When restoring, the Gateway and all deployed subsystems (Management, Analytics, and Developer Portal) must be at the same version level.

The default Developer Portal backup schedule is once every 24 hours, but the schedule can be changed in the backup settings. The Developer Portal saves all system and site backups locally, and also saves them remotely based on the configured SFTP and s3 settings.

The local backups are automatically maintained so that the latest three backups of each site and of the system are kept, and older backups are removed. This maintenance means that the Developer Portal retains the latest three backups for each site and for the system however old they are, but there is no deletion of the old backups on the remote server. If a site is deleted, then all of the local backups for that site are also deleted, as otherwise the backup volume might become full of old site backups. For remote backups, you can configure a retention policy on your remote server to remove the old backup files as required.

- To complete a backup and restore of your Portal Subsystem, you must use `apicup` to configure the backup settings.
- Backups can be scheduled or generated manually (on-demand).
- You can then list the available backups are available and select one to restore your Portal Subsystem when necessary.
- Supported backup types:

- Cloud-based S3 storage
 - IBM Cloud Object Storage
 - AWS S3
- SFTP
- [Configuring backup settings for the Developer Portal subsystem](#)
You can use the `apicup` utility to configure backup settings for the Developer Portal subsystem
- [Backing up the Developer Portal subsystem](#)
After you have configured backup settings, you can use `apicup` to create a manual backup of the Developer Portal.
- [Restoring the Developer Portal subsystem](#)
You can use `apicup` to restore the Developer Portal subsystem.

Configuring backup settings for the Developer Portal subsystem

You can use the `apicup` utility to configure backup settings for the Developer Portal subsystem

Before you begin

Review [Backing up and restoring the Developer Portal](#).

About this task

- During an initial installation, use these steps to configure the backup settings prior to creating the initial management ISO.
- You can configure scheduled backups, or generate manual (on-demand) backups.
- **Version 10.0.3.0 or later:** OpenSSH key authentication for SFTP is supported.

The default Developer Portal backup schedule is once every 24 hours, but the schedule can be changed in the backup settings. The Developer Portal saves all system and site backups locally, and also saves them remotely based on the configured SFTP and s3 settings.

The local backups are automatically maintained so that the latest three backups of each site and of the system are kept, and older backups are removed. This maintenance means that the Developer Portal retains the latest three backups for each site and for the system however old they are, but there is no deletion of the old backups on the remote server. If a site is deleted, then all of the local backups for that site are also deleted, as otherwise the backup volume might become full of old site backups. For remote backups, you can configure a retention policy on your remote server to remove the old backup files as required.

Tip: If you are using remote backups, and there are a large number of rows in your `portalbackup` list that have a `CR TYPE` of `record`, there might be a slow down in system performance, including during Developer Portal upgrades. Although there is no upper limit to how many portal backups you can have, it is recommended that you prune your remote backups so that the number of rows does not become excessively large. One `portalbackup` row of type `record`, corresponds to one backup file on the configured remote backup server. For more information, see [Overview of the Developer Portal backup resources](#).

Procedure

1. Open your API Connect installation project directory.
2. Run the following commands to set the location and timing of your backups:

```
apicup subsys set <subsystem_name> <setting>=<value>

apicup subsys set pt1 site-backup-host=mybackuphost.com
apicup subsys set pt1 site-backup-port=22
apicup subsys set pt1 site-backup-auth-user=mybackupauthusername
apicup subsys set pt1 site-backup-auth-pass=mybackupauthpassword
apicup subsys set pt1 site-backup-path=/site-backups
apicup subsys set pt1 site-backup-protocol=sftp
apicup subsys set pt1 site-backup-certs=<custom-s3-server-CA-cert>
apicup subsys set pt1 site-backup-s3-uri-style=<host-or-path>
apicup subsys set pt1 site-backup-schedule="0 2 * * *"
apicup subsys set pt1 site-priority-list=portal.a1.example.com/test-1/spacecatalog,portal.a1.example.com/test-4/cat3,portal.a1.example.com/test-3/cat1
```

Use `apicup certs` instead of `apicup subsys` to configure ssh-key authentication for SFTP:

```
apicup certs set pt1 site-backup-auth-ssh-key <ssh-key-file-path>
```

The backup parameters are detailed in the following table.

Table 1. Portal backup parameters

Parameter	Description
<code>site-backup-host</code>	<p>The fully qualified domain name of the backup server, in lowercase only.</p> <ul style="list-style-type: none"> • For <code>objstore</code> type backup, specify the S3 endpoint with the corresponding S3 region in the format <code><S3endpoint></code>, or with an optional S3 region in the format <code><S3endpoint>/<S3region></code>. <pre>apicup subsys set pt1 site-backup-host=s3.eu-gb.cloud-object-storage.appdomain.cloud</pre> <ul style="list-style-type: none"> • For <code>sftp</code> type, the SFTP server hostname <pre>apicup subsys set pt1 site-backup-host=my.sftp.backup.domain.example.com</pre>
<code>site-backup-port</code>	<p>The port for the protocol to connect to the <code>site-backup-host</code>. Defaults to 22 if not explicitly set. The backup port is not required for object storage but if specified, is typically port 443.</p> <pre>apicup subsys set pt1 site-backup-port=22</pre>

Parameter	Description
site-backup-auth-user	The user name for the server specified in <code>site-backup-host</code> . If using object storage, the user name is the S3 Secret Key ID. <code>apicup subsys set ptl site-backup-auth-user=<user_name></code>
site-backup-auth-pass	The password for the server specified in <code>site-backup-host</code> . If using object storage, the password is the S3 Secret Access Key parameter. The password is stored in Base64 encoded format, and must not be edited directly in the <code>apiconnect-up.yml</code> file. <code>apicup subsys set ptl site-backup-auth-pass=password</code>
site-backup-auth-ssh-key	Version 10.0.3.0 or later: OpenSSH key for database backups by using SFTP. Important: <ul style="list-style-type: none"> Use <code>apicup certs</code> to set this value. Do not use <code>apicup subsys</code>. Only OpenSSH keys are supported.¹ <code>apicup certs set ptl site-backup-auth-ssh-key=<ssh-key-file-path></code>
site-backup-path	The location of where the backups are stored: <ul style="list-style-type: none"> S3 - the name of your S3 bucket to store backup data, optionally followed by a <code>/</code> and then any subdirectories inside the bucket. Valid examples: <code>bucket</code> <code>bucket/path</code> <code>bucket/lots/of/paths</code> Example: <code>apicup subsys set ptl site-backup-path=apic-backup</code> <ul style="list-style-type: none"> SFTP - the full absolute path of the folder on the SFTP server, beginning with <code>/</code>. <code>apicup subsys set ptl site-backup-path=~/folder></code>
site-backup-protocol	The protocol that is used to communicate with your remote backup endpoint. Specify one of the following values: <ul style="list-style-type: none"> <code>sftp</code> - for secure file transfer protocol <code>local</code> - for local storage <code>objstore</code> - for S3 compatible object storage Starting with v10.0.3.0, the default protocol is <code>local</code> (earlier releases used <code>sftp</code> as the default). For example: <code>apicup subsys set ptl site-backup-protocol=local</code>
site-backup-certs	The name of a custom certificate that contains your upstream custom S3 CA certificate. Supported beginning with 10.0.2. This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be <code>ca.crt</code> and the value is the base64-encoded value of the CA certificate. If the server certificate that is presented by your S3/object-store endpoint is signed by a well known CA and includes any intermediate certificates in the chain, then you do not need to provide the CA certificate using this feature because the Portal subsystem already trusts the server certificate. You can test this by configuring the Portal backup to your S3/object-store server without providing the CA certificate. If you do need to provide the CA certificate then you must provide the entire chain in the <code>ca.crt</code> member of the secret, as follows: <code>intermediate-certificate-1</code> <code>intermediate-certificate-2</code> <code>. . .</code> <code>intermediate-certificate-n</code> <code>root-certificate</code> Where the first certificate in the <code>ca.crt</code> member (intermediate-certificate-1) is the issuer of the S3/object-store server certificate, and each subsequent certificate in the <code>ca.crt</code> member is the issuer of the preceding certificate. The root certificate in the <code>ca.crt</code> member must be self-signed. If there are no intermediate certificates involved, then the <code>ca.crt</code> member contains only the root certificate. The following sample bash script creates a Kubernetes secret: <code>cat >customs3ca.yaml <<EOF</code> <code>apiVersion: v1</code> <code>data:</code> <code> ca.crt: \$(base64 <path-to-ca-certificate> tr -d '\n')</code> <code>kind: Secret</code> <code>metadata:</code> <code> name: custom-server-ca</code> <code>type: generic</code> <code>EOF</code> <code>kubectl apply -f customs3ca.yaml -n <namespace></code>
site-backup-s3-uri-style	Optional. Indicates whether URIs to your S3 backup should use specify a host or a path value. When not specified, defaults to <code>host</code> . Supported beginning with 10.0.2. Valid values: <code>host</code> and <code>path</code> . Some custom S3 providers require URI style to be set to <code>path</code> . For example, MinIO supports both <code>host</code> and <code>path</code> style setup. You can create a MinIO S3 server to only accept <code>path</code> style client communications. Important: Contact your custom S3 administrator before configuring this field. If not properly configured, an upstream custom S3 can reject connections from the client, such as the API Connect Management subsystem.

Parameter	Description
<code>site-backup-schedule</code>	<p>The schedule for how often automatic Portal backups are run. The format for the schedule is any valid cron string, as follows:</p> <pre> * * * * * - - - - - +----- day of week (0 - 6) (Sunday=0) +----- month (1 - 12) +----- day of month (1 - 31) +----- hour (0 - 23) +----- min (0 - 59) </pre> <p>For example: <code>30 22 * * 1</code> performs backups at 10:30 pm on Mondays. The default backup schedule is <code>0 2 * * *</code> (runs every day at 2 am).</p> <p>The timezone for backups is UTC.</p> <pre> apicup subsys set pt1 site-backup-schedule="0 2 * * *" </pre>
<code>site-priority-list</code>	<p>Optional list of prioritized sites to restore. Default: empty.</p> <pre> apicup subsys set <portal-subsystem-name> site-priority-list <comma-separated-list-of-sites> </pre> <p>For example:</p> <pre> apicup subsys set pt1 site-priority-list portal.a1.example.com/e2etest-1/spacecatalog,portal.a1.example.com/e2etest-4/cat3,portal.a1.example.com/e2etest-3/cat1 </pre>

3. At any time you can view the current Portal subsystem values:

- For most values:

```
apicup subsys get pt1
```

where `pt1` is the name that you assigned to your Portal service. The output from this command lists all of the subsystem settings, including backup, and indicates whether there are any errors. You must fix any errors before continuing.

- For ssh-key authentication for SFTP:

```
apicup certs get pt1 site-backup-auth-ssh-key
```

4. Activate the backup settings by running the following command:

```
apicup subsys install pt1
```

where `pt1` is the name that you assigned to your Portal service.

Note: The `apicup subsys install` command automatically runs `apicup health-check` prior to installing the new configuration. An error is displayed if a problem is found that will prevent successful reconfiguration.

In some scenarios, after you fix the problem, the health check might continue to fail. In this case, you can use the `--skip-health-check` flag to suppress the automatic health check:

```
apicup subsys install <subsystem_name> --skip-health-check
```

5. Validate the installation with the new backup parameters by running the following command:

```
apicup subsys get pt1 --validate
```

where `pt1` is the name that you assigned to your Portal service. The output from this command lists all of the subsystem settings, including backup, and indicates whether the values are valid or invalid. You must correct any invalid values before continuing.

6. Ensure that the changes were applied successfully:

```
apicup subsys health-check <subsystem_name>
```

- [Overview of the Developer Portal backup resources](#)

How to manage your Developer Portal remote backup resources.

¹ PuTTY style keys can be converted to OpenSSH by using the PuTTY Key Generator (PuTTYgen) application; see <https://www.puttygen.com/>.

Overview of the Developer Portal backup resources

How to manage your Developer Portal remote backup resources.

Note: This information is relevant only if you're taking remote Developer Portal backups.

The Developer Portal uses two types of portal backup resources. When you run the command `kubect1 get portalbackup`, you see a list of backups, for example:

NAME	ID	TASK ID	STATUS	TYPE	CR TYPE	CLUSTER	AGE	COMMENT
<code>my-backup-f91ms</code>	n/a	20b47fe090f5	Ready	system	create	portal	78m	
<code>portal-record-222cg</code>	20220108.010033		Ready	system	record	portal	94m	
<code>portal-record-22xdc</code>	20220702.010339		Ready	site	record	portal	9m49s	
<code>portal-record-245j7</code>	20221002.010340		Ready	site	record	portal	5m	
<code>portal-record-246hm</code>	20221218.010103		Ready	site	record	portal	32m	
<code>portal-record-24h44</code>	20220509.010037		Ready	site	record	portal	68m	
<code>portal-record-24wb4</code>	20221003.010029		Ready	site	record	portal	60m	
<code>portal-record-264mh</code>	20221119.010453		Ready	system	record	portal	78m	

There are two types of resources listed, and these are denoted by the `CR TYPE` column; `create` or `record` types.

1. **create**: This resource is a representation of a `portalbackup` task that was created manually. See [Backing up and restoring the Developer Portal in a Kubernetes environment](#), [Backing up Developer Portal on OpenShift and Cloud Pak for Integration](#), or [Backing up the Developer Portal subsystem](#) on VMware, for information about how to create a manual portal backup.
2. **record**: This resource is generated by the operator, and it's a representation of a backup file that's stored on the remote backup server. A record listing can be modified only to add a comment to it. You can describe each record to understand which file it is referring to by running the following command:

```
kubectl describe portalbackup portal-record-name
```

For example:

```
kubectl describe portalbackup portal-record-zmmfw
.
.
Name:          portal-record-zmmfw
Namespace:    apic
Labels:       createdBy
.
.
Full Backup Name:  my-portal-s3-hostname@my-porg@catal-20220604.010302.tar.gz
```

Record rows are generated and updated on a schedule, according to the backups that exist on your remote backup server. If you have 10 backup files stored on your remote server, then there will be 10 record rows; one row for each file.

The Developer Portal doesn't have a retention policy for remote backups. Therefore, it's your responsibility to ensure that you're keeping the number of backups that are stored remotely under control. An excessive number of portal backups results in delays administrating and upgrading the Developer Portal. This is due to the operator needing to maintain the excessive number of record rows that correspond to each remote portal backup.

Backing up the Developer Portal subsystem

After you have configured backup settings, you can use `apicup` to create a manual backup of the Developer Portal.

About this task

You can make on-demand backups of your Developer Portal system, sites, and complete service, by running the following `apicup` commands in your API Connect installation project directory.

The default Developer Portal backup schedule is once every 24 hours, but the schedule can be changed in the backup settings. The Developer Portal saves all system and site backups locally, and also saves them remotely based on the configured SFTP and s3 settings.

The local backups are automatically maintained so that the latest three backups of each site and of the system are kept, and older backups are removed. This maintenance means that the Developer Portal retains the latest three backups for each site and for the system however old they are, but there is no deletion of the old backups on the remote server. If a site is deleted, then all of the local backups for that site are also deleted, as otherwise the backup volume might become full of old site backups. For remote backups, you can configure a retention policy on your remote server to remove the old backup files as required.

Tip: If you are using remote backups, and there are a large number of rows in your `portalbackup` list that have a `CR TYPE` of `record`, there might be a slow down in system performance, including during Developer Portal upgrades. Although there is no upper limit to how many portal backups you can have, it is recommended that you prune your remote backups so that the number of rows does not become excessively large. One `portalbackup` row of type `record`, corresponds to one backup file on the configured remote backup server. For more information, see [Overview of the Developer Portal backup resources](#).

Procedure

1. Ensure your system is healthy:

```
apicup subsys health-check [subsystem_name]
```

When successful, the command returns silently with an exit code of 0.

2. Create a backup:

```
apicup subsys backup [subsystem_name] [flags]
```

- To backup the Portal system configuration (and not the sites), run the following command:

```
apicup subsys backup pt1 --type system
```

- To backup a specific Portal site or all sites, run the following command:

```
apicup subsys backup pt1 --type site --site-name arg
```

where *arg* is a specific site UUID or URL, or to backup all sites use the argument `installed`.

- To backup the entire Portal service (the system and all installed sites), run the following command:

```
apicup subsys backup pt1
```

Where *pt1* is the name that you assigned to your Portal service.

Note: To restore a Developer Portal service, you need backups of both the Portal system and the installed sites.

To view additional flags, use `-h`:

```
apicup subsys backup portal -h
Backup portal subsystem
```

```
Usage:
  backup SUBSYS [flags]
```



```

Flags:
--comment string      Comment to add to the backup CR before creation
--debug              Enable debug logging
-h, --help            help for backup
--site-name string    Site to restore to the remote server if TYPE=site. Accepts site uuid/url or 'installed' to
backup all installed sites.
--type string         Type of backup to take ['system' (backup Portal System), 'site' (backup Portal Site),
'all' (complete backup)] (default "all")
--wait               Wait for the operation to complete or fail.
--wait-timeout duration Command timeout in seconds. (default 40s)

```

3. To view the available backups and their status:

```
apicup subsys list-backups [subsystem_name] [flags]
```

Optional flags:

```

./apicup subsys list-backups -h
List backups of the subsystem
Usage:
  apicup subsys list-backups SUBSYS [flags]
Flags:
  -h, --help            help for list-backups
  --timeout duration    Command timeout in seconds. (default 40s)
Global Flags:
  --accept-license      Accept the license for API Connect
  --debug               Enable debug logging

```

Restoring the Developer Portal subsystem

You can use `apicup` to restore the Developer Portal subsystem.

About this task

You can restore your Developer Portal service by using the backups that exist on your remote server, by running the following `apicup` commands in your API Connect installation project directory. Note that before you can run these commands, you must have configured your remote backup server details, and have valid backups of both the Portal system and the installed sites. See [Configuring backup settings for the Developer Portal subsystem](#) and [Backing up the Developer Portal subsystem](#).

Note:

- If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. The backups of the Management and Portal must be taken at the same time to ensure that the Portal sites are consistent with Management database.
- Restoration requires a functioning Developer Portal. In a disaster recovery scenario, you might need to reinstall the Developer Portal subsystem before you can restore the backed-up data. For disaster recovery, see [Recovering the Developer Portal subsystem on VMware](#).

Procedure

1. To view the available backups and their status:

```
apicup subsys list-backups subsystem_name [flags]
```

Optional flags:

```

./apicup subsys list-backups -h
List backups of the subsystem
Usage:
  apicup subsys list-backups SUBSYS [flags]
Flags:
  -h, --help            help for list-backups
  --timeout duration    Command timeout in seconds. (default 40s)
Global Flags:
  --accept-license      Accept the license for API Connect
  --debug               Enable debug logging

```

2. Review which restore actions will be performed

To see what restore actions are performed when the `apicup subsys restore ptl` command is run, you can use the following command:

```
apicup subsys restore ptl --dry-run --timestamp [now|YYYYMMDD.HHMMSS]
```

where `--timestamp` can be `now`, so the latest backup file that is available is used. Or you can specify a timestamp in the format of `YYYYMMDD.HHMMSS` to retrieve the nearest backup file to a specified time, searching backwards from the timestamp given.

Note:

- From IBM® API Connect Version 10.0.3.0, the timestamp format changed to `YYYYMMDD.HHMMSS`. For Version 10.0.2.0 and earlier, the timestamp format is `YYYY-MM-DD HH:MM:SS`.

- With the `--dry-run` option, the utility runs through the restore steps without executing them. You can use this option to see if the restore from the selected backup will succeed.

The `--dry-run` option starts with the following output message:

```
Starting restore...please check later for completion status of restore
PortalRestore is Running
```

To view the results of the dry-run, you can either look in the portal restore CR for a message of completion (or failure) or you can get a high-level status by entering:

```
apicup subsys list-restores <subsys_name>
```

Status values can be: **Ready**, **Completed**, **Failed** or **Running**.

3. Run the following command, and attach any flags as appropriate.

```
apicup subsys restore [SUBSYS_NAME] [flags]
```

- To restore your Portal service, run the following command:

```
apicup subsys restore ptl --timestamp [now|YYYYMMDD.HHMMSS]
```

where `--timestamp` can be `now`, so the latest backup file that is available is used. Or you can specify a timestamp in the format of `YYYYMMDD.HHMMSS` to retrieve the nearest backup file to a specified time, searching backwards from the timestamp given. This command executes the portal restore process, which downloads the system backup and all of the site backups from the remote server, and installs them within the Portal stack. This process will then restore the system configuration from the found backup, and restore all of the sites. Note that if a backed up site is already installed on the current stack, then the site is reinstalled by using the backup (the site is overwritten by the backup). If there are multiple sites to restore, then these sites are queued for restoring. You can track the restoration process within the Portal `www admin` logs.

- By default the restore is a complete restore. You can optionally include `--type all` on the command line to specify a complete restore, but the flag is not required, as a complete restore is performed by default. Note that a complete restore supports only remote restores. Note that the complete restore uses the `priorityList` configuration setting that you specified in [Configuring backup settings for the Developer Portal subsystem](#).

- To restore just the Portal system configuration, run the following command:

```
apicup subsys restore ptl --type system --system-name arg
```

where `arg` can be a `.tgz` name or also accepts `backup ID` if `TYPE=system` and `CR TYPE=record` from the `list-backups` command.

Note that if Portal system configuration information exists on the current stack, running this command will overwrite that configuration.

- To restore just the Portal sites, run the following command:

```
apicup subsys restore ptl --type site --site-name arg
```

where `arg` can be the site backup `tgz` file name or URL, to restore a particular site by using the specified backup ID (or the latest backup file found if using a URL). Or you can use the argument `all` to restore all of the Portal sites that have backup files on the remote server. Note that if any of the backed up sites are already installed on the current stack, then they are reinstalled by using the backup (the sites are overwritten by the backup).

```
> ./apicup subsys restore <portal> -h
Restore portal subsystem
```

Usage:

```
restore SUBSYS [flags]
```

Flags:

```
--custom-platform-api-hostname string  The custom platform API hostname to override for the portal system when
restoring it. Used only when performing a portal restore system or portal restore all.
--custom-url string                    The custom URL to override for the portal site when restoring it. Used only
when performing a portal site restore.
--debug                                Enable debug logging
--dry-run                               Runs a dry run of the command, returning the actions that will be taken.
-h, --help                             help for restore
--site-name string                     File to restore site from which should exist on the pod's local filesystem
or the remote server. Also accepts backup ID if TYPE=site and CR TYPE=record.
--system-name string                   File to restore system from which should exist on the pod's local filesystem
or the remote server. Also accepts backup ID if TYPE=system and CR TYPE=record.
--timestamp string                     In YYYYMMDD.HHMMSS format, specify a timestamp to retrieve the backup from.
The nearest backup, searching backwards from this timestamp, is used.
--type string                           Type of restore to take ['system' (restore Portal System), 'site' (restore
Portal Site), 'all' (complete restore)] (default "all")
--wait                                  Wait for the operation to complete or fail.
--wait-timeout duration                Command timeout in seconds. (default 40s)
```

4. To view your restores and their statuses, run:

```
apicup subsys list-restores <subsys_name> <flags>
```

Help output:

```
apicup subsys list-restores -h
List restores of the subsystem
```

Usage:

```
apicup subsys list-restores SUBSYS [flags]
```

Flags:

```
-h, --help                help for list-restores
--timeout duration        Command timeout in seconds. (default 40s)
```

Global Flags:

```
--accept-license  Accept the license for API Connect
--debug           Enable debug logging
```

5. To list the Portal backup sites, use the Portal CLI command. See [Using the sites commands](#).

6. To list the Portal system sites, use the Portal CLI command. See [Using the platforms commands](#).

Backing up and restoring the Analytics database on VMware

The IBM® API Connect Analytics database can be backed up and restored from an S3 repository. S3 compatible object storage is required; for example, IBM Cloud Object Storage.

About this task

Begin by configuring your backup configuration, which optionally includes a schedule if you want to automate your backups. You can run backups manually as needed. To restore a backup, review the set of available backups, and then restore the selected backup.

Note: On a v10.0.5 API Connect deployment, you can restore only analytics backups that were taken from a v10.0.5 analytics deployment. You cannot restore analytics backups that were taken from a deployment that was on v10.0.4 or an earlier release.

- [Configuring backup settings for Analytics](#)
Use the `apicup` utility to configure backup settings for the Analytics database in your VMware environment.
- [Running a backup of the Analytics database](#)
Initiate a backup of the API Connect Analytics database on VMware.
- [Restoring the Analytics database](#)
Restore a selected backup of the IBM API Connect Analytics database on VMware.

Configuring backup settings for Analytics

Use the `apicup` utility to configure backup settings for the Analytics database in your VMware environment.

About this task

The Analytics database can be backed up and restored from an S3 repository. S3-compatible object storage is required; for example, IBM Cloud Object Storage.
Note:

- Only host-style S3 backups are supported. Path-style S3 backups are not supported.
- If you are using a self-managed S3 backup server, check that you have a DNS hostname entry and matching TLS certificate for your S3 backup endpoint.

Procedure

1. Use the `apicup` utility to configure the backup settings.
Configure each setting by running the following command with the key and value from Table 1 that you want to apply:

```
apicup subsys set <analytics_subsystem> <key> <value>
```

Table 1 shows the supported backup settings with the default value (if any) and a description of each setting.

Table 1. Analytics backup settings

Key	Default value	Description
<code>analytics-backup-auth-pass</code>		Server password for Analytics backups.
<code>analytics-backup-auth-user</code>		Server user name for Analytics backups.
<code>analytics-backup-host</code>		The S3 endpoint with the corresponding S3 region in the format: <code>s3.s3region.s3domain</code> For example: <code>s3.eu-gb.cloud-object-storage.appdomain.cloud</code> Attention: A path style host value (formatted as <code>S3endpoint/S3region</code>) is not supported for AWS.

Key	Default value	Description
<code>analytics-backup-certs</code>		<p>The name of a custom certificate that contains your upstream custom S3 CA certificate. Supported beginning with 10.0.2. This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be <code>ca.crt</code> and the value is the base64-encoded value of the CA certificate.</p> <p>If the server certificate that is presented by your S3/object-store endpoint is signed by a well known CA and includes any intermediate certificates in the chain, then you do not need to provide the CA certificate using this feature because the Portal subsystem already trusts the server certificate. You can test this by configuring the Portal backup to your S3/object-store server without providing the CA certificate.</p> <p>If you do need to provide the CA certificate then you must provide the entire chain in the <code>ca.crt</code> member of the secret, as follows:</p> <pre>intermediate-certificate-1 intermediate-certificate-2 . . intermediate-certificate-n root-certificate</pre> <p>Where the first certificate in the <code>ca.crt</code> member (<code>intermediate-certificate-1</code>) is the issuer of the S3/object-store server certificate, and each subsequent certificate in the <code>ca.crt</code> member is the issuer of the preceding certificate. The root certificate in the <code>ca.crt</code> member must be self-signed.</p> <p>If there are no intermediate certificates involved, then the <code>ca.crt</code> member contains only the root certificate.</p> <p>The following sample bash script creates a Kubernetes secret:</p> <pre>cat >customs3ca.yaml <<EOF apiVersion: v1 data: ca.crt: \$(base64 <path-to-ca-certificate> tr -d '\n') kind: Secret metadata: name: custom-server-ca type: generic EOF kubect1 apply -f customs3ca.yaml -n <namespace></pre>
<code>analytics-backup-path</code>		A combination of the S3 bucket and the base path within the bucket, using the format: <code>bucket_name/path</code>
<code>analytics-backup-chunk-size</code>	1GB	(Optional) Specifies how large files are stored. Large files can be stored as chunks when the snapshot is created. This setting specifies the size of the chunks as GB, MB, or KB.
<code>analytics-backup-schedule</code>	0 2 ***	(Optional) Determines how often backups will be invoked automatically. The format for the schedule is any valid <code>cron</code> string. The timezone for backups is UTC. Backing up once a day is probably sufficient; remember that the more frequently you run backups, the more storage space you need.
<code>analytics-enable-compression</code>	true	(Optional) Determines whether metadata files are stored in compressed format.
<code>analytics-enable-server-side-encryption</code>	false	(Optional) Determines whether files are encrypted. When set to true, files are encrypted on the server side using AES256.

2. Validate the backup settings by running the following command:

```
apicup subsys get <analytics_subsystem> --validate
```

3. Apply the backup settings to the Analytics subsystem by reinstalling with the following command:

```
apicup subsys install <analytics_subsystem>
```

Note: The `apicup subsys install` command automatically runs `apicup health-check` prior to installing the new configuration. An error is displayed if a problem is found that will prevent successful reconfiguration.

In some scenarios, after you fix the problem, the health check might continue to fail. In this case, you can use the `--skip-health-check` flag to suppress the automatic health check:

```
apicup subsys install <subsystem_name> --skip-health-check
```

4. Verify that the backup settings were successfully applied by running the following command:

```
apicup subsys health-check <analytics_subsystem>
```

Running a backup of the Analytics database

Initiate a backup of the API Connect Analytics database on VMware.

About this task

Procedure

1. Perform a health check of your Analytics system by running the following command:

```
apicup subsys health-check <analytics_subsystem>
```

If successful, the command returns silently with exit code 0.

2. Start the backup by running the following command and including the appropriate flags:

```
apicup subsys backup <analytics_subsystem>
```

View the list of supported flags by running the `backup` command with the `-h` (help) flag, as shown in the following example:

```
apicup subsys backup analytics -h
Backup analytics subsystem
```

Usage:

```
backup SUBSYS [flags]
```

Flags:

```
--comment string      Comment to add to the backup CR before creation
--debug               Enable debug logging
-h, --help            help for backup
--ignore-unavailable If false, the backup will fail if an index is missing. If true, missing indices will be
skipped and the backup will continue.
--indices strings     Indice(s) you'd like to backup (all/apievents/ui). It can be set to any combination of
indices (e.g. ui or ui,apievents).
--wait                Wait for the operation to complete or fail.
--wait-timeout duration Command timeout in seconds. (default 40s)
```

Note: The `--indices` flag can be set to `all|apievents|ui`, or to the names of individual indexes. `all` includes `apievents` and `ui` indexes. Multiple values can be supplied with a comma-separated list. For example, to backup all UI data and all data from August of 2022:

```
--indices ui, "apic-api-2022.08*"
```

3. Optionally view a list of available backups, and the status of each, by running the following command:

```
apicup subsys list-backups <analytics_subsystem>
```

View the list of supported flags by running the `list-backups` command with the `-h` (help) flag, as shown in the following example:

```
apicup subsys list-backups -h
List backups of the subsystem
```

Usage:

```
apicup subsys list-backups SUBSYS [flags]
```

Flags:

```
-h, --help            help for list-backups
--timeout duration   Command timeout in seconds. (default 40s)
```

Global Flags:

```
--accept-license    Accept the license for API Connect
--debug             Enable debug logging
```

Restoring the Analytics database

Restore a selected backup of the IBM® API Connect Analytics database on VMware.

About this task

Use the `apicup` utility to view the list of available backups for the Analytics database, and then restore a selected backup.

Procedure

1. Retrieve a list of available backups, and the status of each, by running the following command:

```
apicup subsys list-backups <analytics_subsystem>
```

View the list of supported flags by running the `list-backups` command with the `-h` (help) flag, as shown in the following example:

```
apicup subsys list-backups -h
List backups of the subsystem
```

Usage:

```
apicup subsys list-backups SUBSYS [flags]
```

Flags:

```
-h, --help            help for list-backups
--timeout duration   Command timeout in seconds. (default 40s)
```

Global Flags:

```
--accept-license    Accept the license for API Connect
--debug             Enable debug logging
```

The response includes the unique ID of each backup. Note the ID of the backup that you want to restore, so you can use it in the next step.

2. Restore a backup by running the following command with the appropriate flags:

```
apicup subsys restore <analytics_subsystem>
```

View the complete list of supported flags by running the `restore` command with the `-h` (help) flag, as shown in the following example:

```
apicup subsys restore analytics -h
Restore analytics subsystem
```

Usage:

```
restore SUBSYS [flags]
```

Flags:

```
--debug          Enable debug logging
-h, --help       help for restore
--id string      ID of the backup to restore
--ignore-unavailable If false, the restore will fail if an index is missing. If true, missing indices will be
skipped and the restore will continue.
--indices strings Indice(s) you'd like to restore (all/apievents/ui). It can be set to any combination of
indice(s) (eg. ui or ui,apievents) (default [all])
--override       If set to true, then the restore operation will override existing indices.
--wait           Wait for the operation to complete or fail.
--wait-timeout duration Command timeout in seconds. (default 40s)
```

- `--id` the backup ID provided must be enclosed in double quotes.
- `--indices` can be set to `all|apievents|ui`, or to the names of individual indices. `all` includes `apievents` and `ui` indices. Multiple values can be supplied with a comma-separated list. For example, to restore all UI data and all data from August of 2022:

```
--indices "ui,apic-api-2022.08"
```
- `--override` by default is set to `false` to prevent you from accidentally replacing existing indices. If `--override` is set to `false` and you specify indices that already exist in the database that you are restoring, then the restore fails. The failure prevents you from accidentally replacing or losing data that exists in the current database.

3. Optionally view a list of available restores, and the status of each, by running the following command:

```
apicup subsys list-restores <analytics_subsystem>
```

View the list of supported flags by running the `list-restores` command with the `-h` (help) flag, as shown in the following example:

```
apicup subsys list-restores -h
List restores of the subsystem
```

Usage:

```
apicup subsys list-restores SUBSYS [flags]
```

Flags:

```
-h, --help          help for list-restores
--timeout duration Command timeout in seconds. (default 40s)
```

Global Flags:

```
--accept-license Accept the license for API Connect
--debug          Enable debug logging
```

Important:

For deployment profiles with fewer than 3 storage nodes, the restore process is never marked as **Complete**. The storage cluster status is expected to be yellow and to have more than 0 unassigned shards. In this situation, the restore process is likely to be complete when there are 0 initializing shards.

Disaster recovery

Prepare for disaster events and recover API Connect on VMware if a disaster event occurs.

You must complete preparation steps prior to a disaster in order to recover API Connect on VMware.

The backups that you create for disaster recovery are for recovery of the API Connect subsystems in the same environment that the backups were taken, or in a different environment that has the same network configuration and API Connect form factor. If you want to move your API Connect deployment to a different environment and change the form factor or modify your API Connect endpoints, then see [Migrating from v10 to v10 on a different form factor](#).

Note: For information about how to maintain a two data center deployment, including failover and recovery procedures, see [Maintaining a two data center deployment](#).

- [Preparing for a disaster](#)
Prepare your API Connect deployment for disaster recovery in a VMware environment.
- [Recovering from a disaster](#)
Recover the API Connect subsystems on VMware.
- [Using VM snapshots for infrastructure backup and disaster recovery](#)
You can use VM snapshots to backup and restore the infrastructure used by API Connect on VMware, and also for infrastructure disaster recovery.

Preparing for a disaster

Prepare your API Connect deployment for disaster recovery in a VMware environment.

Important: Successful disaster recovery of API Connect depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery.

- [Preparing the management subsystem for disaster recovery on VMware](#)
Prepare a Management subsystem for disaster recovery by taking specific steps before and after the installation of the subsystem.
- [Preparing the Developer Portal subsystem for disaster recovery on VMware](#)
Prepare a Developer Portal subsystem for disaster recovery by taking specific steps immediately after the installation of the subsystem.
- [Preparing the Analytics subsystem for disaster recovery on VMware](#)
Prepare the API Connect Analytics subsystem for disaster recovery in a VMware environment.

Preparing the management subsystem for disaster recovery on VMware

Prepare a Management subsystem for disaster recovery by taking specific steps before and after the installation of the subsystem.

About this task

- This task must be performed before any disaster event occurs, and also prior to the installation of a replacement Management subsystem during the recovery process. Best practice is to complete these steps immediately after initial configuration of a management subsystem during your original v10 deployment.
- Any **local** backups are presumed to have been lost in the disaster scenario and are non-recoverable in this procedure.

Important: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

Procedure

1. Make sure that you have a backup that can be used in case of a disaster event:
 - For backup configuration, see [Configuring backup settings during initial installation of the management subsystem](#).
 - If scheduled backups were configured during backup configuration, you should have access to a backup on the specified cloud storage.
 - You can perform a manual backup at any time. See [Backing up the management subsystem](#).
2. Ensure that you have a backup of your project directory.

The original project directory that was created with the `apicup` command during the initial product installation is required for disaster recovery, see: [First steps for deploying in a VMware environment](#). The project directory contains the yaml files that describe your deployment, encryption keys, and deployment ISO files. It is not possible to restore the subsystem databases on a deployment that uses a new or different project directory.
3. Optional: Take a Virtual Machine (VM) snapshot of all your VMs; see [Using VM snapshots for infrastructure backup and disaster recovery](#) for details. This action does require a brief outage while all of the VMs in the subsystem cluster are shut down - do not take snapshots of running VMs, as they might not restore successfully. VM snapshots can offer a faster recovery when compared to redeploying OVA's and restoring from normal backups.

Important: VM snapshots are not an alternative to the standard API Connect backups that are described in the previous step. Do not rely solely on VM snapshots for your backups.

What to do next

You should now complete the preparation steps for the Developer Portal subsystem; see [Preparing the Developer Portal subsystem for disaster recovery on VMware](#).

Preparing the Developer Portal subsystem for disaster recovery on VMware

Prepare a Developer Portal subsystem for disaster recovery by taking specific steps immediately after the installation of the subsystem.

About this task

- Perform this task before any disaster event occurs, and also prior to the installation of a replacement Developer Portal subsystem for the recovery process.
- Best practice is to complete these steps immediately after initial configuration of a Developer Portal subsystem during your original v10 deployment.
- Any **local** backups are presumed to have been lost in the disaster scenario and are non-recoverable in this procedure.
- Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. For information about preparing the Management subsystem, see [Preparing the management subsystem for disaster recovery on VMware](#).

If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

Procedure

1. Before the initial installation, configure the Developer Portal subsystem for database backups. See [Configuring backup settings for the Developer Portal subsystem](#).
2. Ensure you create a backup:
 - You can perform a manual backup at any time. See [Backing up the Developer Portal subsystem](#).
 - If you configure scheduled backups as part of [Configuring backup settings for the Developer Portal subsystem](#), backups to the specified cloud storage take place at the configured time.
3. Ensure that you have a backup of your project directory.

The original project directory that was created with the `apicup` command during the initial product installation is required for disaster recovery, see: [First steps for deploying in a VMware environment](#). The project directory contains the yaml files that describe your deployment, encryption keys, and deployment ISO files. It is not possible to restore the subsystem databases on a deployment that uses a new or different project directory.
4. Optional: Take a Virtual Machine (VM) snapshot of all your VMs; see [Using VM snapshots for infrastructure backup and disaster recovery](#) for details. This action does require a brief outage while all of the VMs in the subsystem cluster are shut down - do not take snapshots of running VMs, as they might not restore successfully. VM snapshots can offer a faster recovery when compared to redeploying OVA's and restoring from normal backups.

Important: VM snapshots are not an alternative to the standard API Connect backups that are described in the previous step. Do not rely solely on VM snapshots for your backups.

What to do next

To restore the Developer Portal on VMware after a disaster, see [Recovering the Developer Portal subsystem on VMware](#).

Preparing the Analytics subsystem for disaster recovery on VMware

Prepare the API Connect Analytics subsystem for disaster recovery in a VMware environment.

About this task

Preparing for disaster recovery involves backing up analytics data and your project directory.

Procedure

1. [Perform a backup of the Analytics database.](#)
Scheduling automatic backups ensures that a recovery includes recent updates. For more information on configuring backups, see [Configuring backup settings for Analytics.](#)
2. Ensure that you have a backup of your project directory.
The original project directory that was created with the `apicup` command during the initial product installation is required for disaster recovery, see: [First steps for deploying in a VMware environment.](#) The project directory contains the yaml files that describe your deployment, encryption keys, and deployment ISO files. It is not possible to restore the subsystem databases on a deployment that uses a new or different project directory.
3. Ensure that you have a backup of the `analytics-extra-values.yaml` file if you created one: [Creating the analytics-extra-values file.](#) This file might be included in the project directory backup if you kept it there.
4. Optional: Take a Virtual Machine (VM) snapshot of all your VMs; see [Using VM snapshots for infrastructure backup and disaster recovery](#) for details. This action does require a brief outage while all of the VMs in the subsystem cluster are shut down - do not take snapshots of running VMs, as they might not restore successfully. VM snapshots can offer a faster recovery when compared to redeploying OVAs and restoring from normal backups.
Important: VM snapshots are not an alternative to the standard API Connect backups that are described in the previous step. Do not rely solely on VM snapshots for your backups.

Recovering from a disaster

Recover the API Connect subsystems on VMware.

- [Recovering the Management subsystem on VMware](#)
Recover the management subsystem from backups after a disaster event.
- [Recovering the Developer Portal subsystem on VMware](#)
Recover the Developer Portal on VMware.
- [Recovering the Analytics subsystem on VMware](#)
Recover the API Connect Analytics subsystem in a VMware environment.

Recovering the Management subsystem on VMware

Recover the management subsystem from backups after a disaster event.

Before you begin

To successfully recover the management subsystem, you must have previously completed the steps in [Preparing the management subsystem for disaster recovery on VMware.](#)

You must use the same project directory that you used for your original deployment, or a restore of your project directory backup, to ensure that configuration and secret information is transferred to the replacement deployment.

In a clustered deployment, if any one VM is corrupted then all of the VMs in the cluster must be redeployed. You cannot replace just a single corrupted VM in a cluster.

Important: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

Procedure

1. Determine which backup to restore from.
 - a. Obtain a list of the available backups for your backup type:
 - s3 backups
 - For IBM®'s Cloud Object Storage, you can check currently stored backups in the COS console. Select Buckets > Objects. See the Object Names displayed on the `<cos_name>/backup/db` panel. For example:

```
20200605-105429F
20200605-100008F
20200605-11040F
```
 - For Amazon's AWS, you can check currently stored backups in the S3 console For example, under Amazon S3 > `cluster_name` > old cluster > backup > db:

```
20200606-144315F
20200606-145011F
```


- SFTP backups

View the SFTP backups available on your remote storage site:

```
-rw-r--r-- 1 root root 13092333 Aug 26 08:56 20200826-154646F.tgz
-rw-r--r-- 1 root root 18703758 Aug 26 09:10 20200826-160010F.tgz
-rw-r--r-- 1 root root 24318561 Aug 26 09:21 20200826-161301F.tgz
```

- Select the backup ID of the backup you want to restore. For example, in the sample SFTP backup list, for the Aug 26 09:21 backup, the backup ID is **20200826-161301F**.

Each filename contains the date, time, and type of the backup stored. The format of the backup ID is **YYYYMMDD-HHMMSS<F|I>**. Full backups are denoted with a suffix **F** on the ID. Incremental backups are denoted with a suffix **I** on the ID. For incremental backups, ensure each incremental backup has its prior full backup also present in storage. You can check this by examining the ID **<prior-backup-id>_<backup-id>**.

Note:

During the disaster recovery process, the S3 configuration detail of the older management system is used, but the older management system must be in offline mode. The old subsystem must be offline because you cannot have two management systems simultaneously using the same s3 bucket name in the database backup configurations.

- Make sure you know the management database cluster name. Use the following steps applicable to your backup type:

Backup type	How to obtain database cluster name
S3	<ol style="list-style-type: none"> Open your IBM Cloud® Object Storage or AWS S3 console and proceed to the bucket location where the old Management subsystem backups are located. Download <code>backup/db/<backup-id>/pg_data/postgresql.conf.gz</code>. Open <code>postgresql.conf</code> to view the database cluster name: <ul style="list-style-type: none"> IBM Cloud Object Storage <pre>cluster_name = 'm1-a6287572-postgres'</pre> <ul style="list-style-type: none"> m1 - Management subsystem name a6287572 - site name AWS S3 <pre>cluster_name = 'm1-c91dc0b9-postgres'</pre> <ul style="list-style-type: none"> m1 - Management subsystem name c91dc0b9 - site name
SFTP	<ol style="list-style-type: none"> Recover the Management database cluster name and siteName by examining the SFTP backup tar. Download or move the SFTP backup tar file and decompress (untar) it. Open <code><management-subsystem-name>-<siteName>-postgres-backrest-shared-repo/backup/db/<backup-id>/pg_data/postgresql.conf.gz</code> which contains the management subsystem name and siteName. For example: <pre># Do not edit this file manually! # It will be overwritten by Patroni! include 'postgresql.base.conf' archive_command = 'source /opt/cpm/bin/pgbackrest/pgbackrest-set-env.sh && pgbackrest archive-push "%p"' archive_mode = 'True' archive_timeout = '60' autovacuum_vacuum_cost_limit = '1000' autovacuum_vacuum_scale_factor = '0.01' cluster_name = 'm1-f785a3e3-postgres'</pre> <p>In this example:</p> <ul style="list-style-type: none"> cluster_name has both the management subsystem name and siteName m1 - Management subsystem name f785a3e3 - site name

- Use your prior existing project directory, or a restore of your project directory backup, to install the Management subsystem:

- Create your ISO files

```
apicup subsys install mgmt --out mgmtplan-out
```

The `--out` parameter and value are required.

In this example, the ISO files are created in the `myProject/mgmtplan-out` directory.

Note: If your original ISO files are still available and you haven't upgraded from the original installation, you can reuse them. However, if you have upgraded your original deployment, you must create new ISO files using the version of `apicup` that corresponds to the version your API Connect installation was on at the time of the disaster. For example, do not attempt to deploy v10.0.5.1 OVAs with ISO files that were created with `apicup` v10.0.4.0.

- Deploy the files into the replacement VMs. See [Deploying the Management subsystem OVA file](#).
- Verify the deployment. See [Verify installation of the Management subsystem](#).

Important:

For S3, the recovery remains in an intermediate state until the restore is complete, and Postgres wal files might cause serious disk issues. To avoid this possibility, continue immediately with the next step.

Note that if you delay completion of the restore:

- Health check might fail. In this case, you can still proceed to the next step and perform a restore.
- Postgres wal files might cause problems by consuming all disk space. In this case, you must either:
 - Re-install the system, prepare again for disaster recovery, and perform the restore.
 - Or increase disk space so that the system returns to a stable state, and then proceed with the restore.

- Once your Management subsystem is ready, confirm the backup ID noted in Step 1 is present on the sftp or s3 server.

- After a few moments, confirm there is a **ManagementBackup** of type **record** and its backup ID matches with the backup ID noted in Step 1.

You can list the management backups using:

```
apicup subsys list-backups <subsystem_name>
```

For example:

NAME	STATUS	ID	CLUSTER	SUBSYSTEM	TYPE	CR TYPE	AGE
mgmt-backup-8hqgg	Ready	20200826-161301F	management-82b290a2-postgres	management	full	record	40s

6. Perform a Management Restore using the name of the backup that has the ID you want to restore.
For example, for ID `20200826-161301F` the backup name is `mgmt-backup-8hqgg`.

For instructions on how to restore, see [Restoring the management subsystem](#).

Once the Management Restore has completed and the database is running again, the data of the old Management subsystem will be successfully restored onto the new Management subsystem. Manual and scheduled backups should perform as normal once again.

What to do next

You should now complete the recovery steps for the Developer Portal subsystem on VMware, see [Recovering the Developer Portal subsystem on VMware](#).

Recovering the Developer Portal subsystem on VMware

Recover the Developer Portal on VMware.

Before you begin

Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. If you must recover a deployment due to a disaster, you must first complete [Recovering the Management subsystem on VMware](#), and then immediately recover the Developer Portal.

Recovery of the Portal subsystem is also dependent on prior completion of the preparation steps: [Preparing the Developer Portal subsystem for disaster recovery on VMware](#)

About this task

- In a disaster recovery scenario, an installation of a new Developer Portal subsystem is required.
- In a clustered deployment, if any one VM is corrupted then all of the VMs in the cluster must be redeployed. You cannot replace just a single corrupted VM in a cluster.
- You must use the same project directory that you used for your original Portal deployment, or a restore of your project directory backup, to ensure that configuration and secret information is transferred to the replacement deployment. The `apiconnect-up-v10.yaml` file contains Portal configuration.

Use the following steps to recover the data from your previous Developer Portal subsystem.

Procedure

1. Use your prior existing project directory, or a restore of your project directory backup, to install the replacement Portal subsystem:
 - a. Create your ISO files.

```
apicup subsys install portal-subsystem-name --out portal-subsystem-nameplan-out
```

The `--out` parameter and value are required.

In this example, the ISO files are created in the `myProject/portal-subsystem-nameplan-out` directory.

Note: If your original ISO files are still available and you haven't upgraded from the original installation, you can reuse them. However, if you have upgraded your original deployment, you must create new ISO files using the version of `apicup` that corresponds to the version your API Connect installation was on at the time of the disaster. For example, do not attempt to deploy v10.0.5.1 OVAs with ISO files that were created with `apicup` v10.0.4.0.

- b. Deploy the files into the replacement VMs. See [Deploying the Developer Portal subsystem OVA file](#).
 - c. Verify the deployment. See [Verifying deployment of the Developer Portal subsystem](#).
2. View the available backups and their status:

```
apicup subsys list-backups [subsystem_name] [flags]
```

Optional flags:

```
./apicup subsys list-backups -h
List backups of the subsystem
Usage:
  apicup subsys list-backups SUBSYS [flags]
Flags:
  -h, --help                help for list-backups
  --timeout duration        Command timeout in seconds. (default 40s)
Global Flags:
  --accept-license          Accept the license for API Connect
  --debug                   Enable debug logging
```

3. Perform a Developer Portal restore.
Run the following command, and attach any flags as appropriate.

```
apicup subsys restore [SUBSYS_NAME] [flags]
```

- To restore your Portal service, run the following command:

```
apicup subsys restore ptl --timestamp [now|YYYYMMDD.HHMMSS]
```

where `--timestamp` can be `now`, so the latest backup file that is available is used. Or you can specify a timestamp in the format of `YYYYMMDD.HHMMSS` to retrieve the nearest backup file to a specified time, searching backwards from the timestamp given. This command executes the portal restore process, which downloads the system backup and all of the site backups from the remote server, and installs them within the Portal stack. This process will then restore the system configuration from the found backup, and restore all of the sites. Note that if a backed up site is already installed on the current stack, then the site is reinstalled by using the backup (the site is overwritten by the backup). If there are multiple sites to restore, then these sites are queued for restoring. You can track the restoration process within the Portal `www admin` logs.

Note: From IBM® API Connect Version 10.0.3.0, the timestamp format changed to `YYYYMMDD.HHMMSS`. For Version 10.0.2.0 and earlier, the timestamp format is `YYYY-MM-DD HH:MM:SS`.

- For a disaster recovery scenario, it is advised that you restore all sites and systems. By default the restore is a complete restore. You can optionally include `-type all` on the command line to specify a complete restore, but the flag is not required, as a complete restore is performed by default. Note that the complete restore uses the `priorityList` configuration setting that you specified in [Configuring backup settings for the Developer Portal subsystem](#).
- For more information, see [Restoring the Developer Portal subsystem](#)

Recovering the Analytics subsystem on VMware

Recover the API Connect Analytics subsystem in a VMware environment.

About this task

Recovering the Analytics subsystem involves deploying a new installation of the Analytics subsystem and then restoring the Analytics database from a back-up copy.

You must use the same project directory that you used for your original deployment, or a restore of your project directory backup, to ensure that configuration and secret information is transferred to the replacement deployment.

In a clustered deployment, if any one VM is corrupted then all of the VMs in the cluster must be redeployed. You cannot replace just a single corrupted VM in a cluster.

Procedure

1. Use your prior existing project directory, or a restore of your project directory backup, to install the replacement Analytics subsystem:
 - a. Create your ISO files:

```
apicup subsys install install-subsystem-name --out analytplan-out
```

The `--out` parameter and value are required.

In this example, the ISO files are created in the `myProject/analytplan-out` directory.

Note: If your original ISO files are still available and you haven't upgraded from the original installation, you can reuse them. However, if you have upgraded your original deployment, you must create new ISO files using the version of `apicup` that corresponds to the version your API Connect installation was on at the time of the disaster. For example, do not attempt to deploy v10.0.5.1 OVAs with ISO files that were created with `apicup` v10.0.4.0.

- b. Deploy the files into the replacement VMs. See [Deploying the Analytics subsystem OVA file](#).
 - c. Verify the deployment. See [Verifying deployment of the Analytics subsystem](#).
2. Restore the Analytics data from backup, as explained in [Restoring the Analytics database](#).

Using VM snapshots for infrastructure backup and disaster recovery

You can use VM snapshots to backup and restore the infrastructure used by API Connect on VMware, and also for infrastructure disaster recovery.

Backup and restore procedures for the databases used by IBM API Connect on VMware are described in [Backing up and restoring on VMware](#). The procedures back up all of the state required by API Connect but do not include the underlying infrastructure software or other internal state.

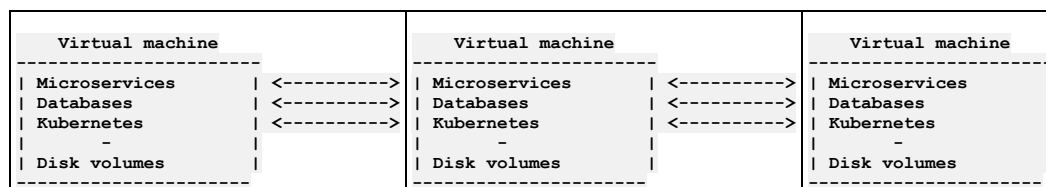
In some scenarios, you may want to perform backups and disaster recovery at an infrastructure layer. You can do this by taking snapshots of the virtual machines or underlying storage volumes, providing that you follow the constraints described here.

Note that performing backups at the infrastructure layer differs from the standard approach in that not only is the database state backed up, but also the precise state of the software. This approach can be useful, for example, when testing upgrades of production systems.

VM snapshots should not be used as an alternative to taking normal backups. Ideal times to take VM snapshots are after initial install and configuration, and before and after upgrades. In the event of a disaster, reverting to the last snapshot taken and then restoring the standard daily backup can be faster than redeploying OVAs and restoring the backup.

- Requirements for taking a consistent backup of an API connect system at the infrastructure level
 - An API Connect system can be approximated by the following diagram:

Table 1. API Connect Installation



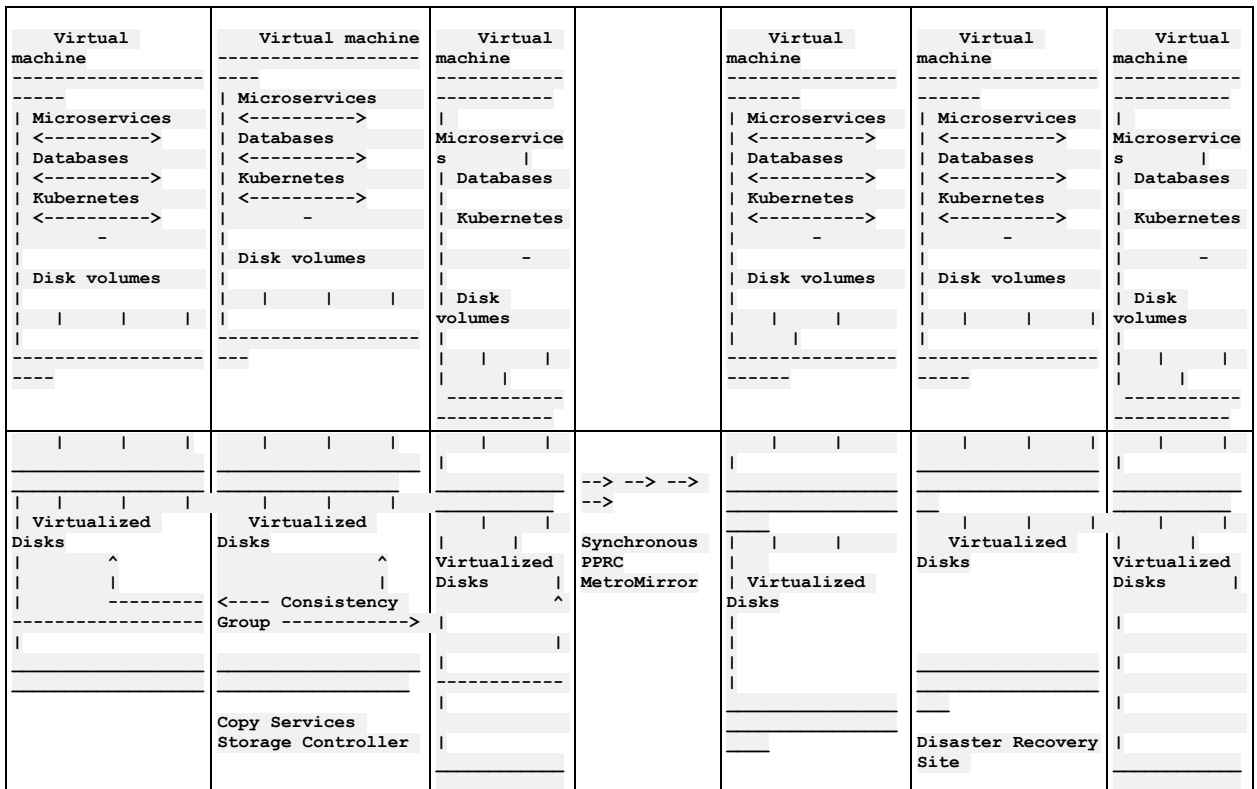
- The system comprises multiple Virtual Machine images which are running Kubernetes, multiple databases, with the API Connect micro services running on top. There is transactional communication between the virtual machines which is occurring at all levels of this stack. This communication occurs all the time, even if the system is seemingly idle.
- Within the system there are multiple stateful or database layers on the software that maintain a consistency protocol. For example `etcd` upon which Kubernetes is based uses the [Raft](#) consensus algorithm. These algorithms depend for consistency on the basic assertion that time flows forwards on all systems together. If one or more of the virtual machine's state were to move backwards relative to the others by even the smallest quantum of time then consistency would be lost and the system might behave badly.

Clearly, if Virtual Machine snapshots are taken across multiple virtual machines it is certain that these snapshots will not be taken at precisely the same time. The result will be a backup that contains a system state that contains data from snapshots taken at slightly different times. It is possible that such a backup system may be restored and may appear to work correctly but hidden deep within it there may be undetected inconsistencies that will cause strange unpredictable errors and inconsistencies later. These issues can be extremely difficult to diagnose.

It is also possible that you can test this procedure successfully multiple times and see no apparent problem occurring. Irrespective of the apparent success of tests, corruption can be occurring in the system state. For this reason, taking VM snapshots or clones of a running API Connect system is not supported.

- How to take a consistent backup of an API connect system at the infrastructure level
 - The only way to take a consistent backup of an API connect system using VM snapshots and clones is to shut down ALL of the virtual machines comprising the system before taking the snapshot or clone.
 - Once the virtual machines are stopped at the VM level, time is effectively frozen. Then a snapshot can be taken on all the VMs. A clone can be made from the snapshot taken on each VM. This set of clones represent a valid, consistent state that the API Connect system was in when it was shut down.
 - Once all of the snapshots and clones have been taken the original API Connect system can be restarted.
- Restoring an API Connect System from VM clones
 - The Cloned VMs represent a valid state of the original API Connect system. They will restart in exactly the same way as the original system restarted.
 - If the objective is to stand up another instance of the cloned API connect system for testing or DR purposes, the following must be true:
 - The original system and the cloned system must be isolated from one another. The clone will be using the same hostnames and IP addresses as the original and so must be stood up in a separate virtual machine hosting environment with Network Address Translation between itself and any network to which the original system is stood up.
 - The cloned system must be stood up in an identical environment in terms of its hardware, software and network.
- Disaster Recovery approaches using Disk Replication:
 - A common DR approach used by enterprises is to use disk based replication. In this case, a storage controller such as IBM's San Volume Controller provides [Copy Services](#). You can use Copy Services to create a completely consistent copy of a set of disks. IBM calls this feature *Synchronous Remote Copy* or *MetroMirror*. Other storage vendors have similar features.
 - If you use Remote Copy to synchronize disk images to a DR site then IBM will support API connect used with these systems so long as the copy at the DR site is completely consistent. Ensuring this consistency is the responsibility of the storage controller and its configuration. It is **critically important** to ensure that ALL of the disk volumes for all of the VMs or nodes in the entire API Connect system are in the same [remote copy consistency group](#) as shown below. This ensures that DR scenarios do not create inconsistent volumes. If the disks are not in the same consistency group then the copy will not be managed consistently and the system will not be supported.

Table 2. API Connect Installation and corresponding Disaster Recovery site



- Infrastructure based backup of API Connect using vSphere
 - The following procedure outlines steps to clone a VMware instance of API Connect v10, using the following topology:
 - 3 Management Nodes
 - 3 Portal Nodes
 - 3 Analytics Nodes
 - 3 Gateways

- Each of these components has a load balancer with its own IP address. The load balancers all sit on a single VM. The cluster is installed and configured on 13 VMs in VMware. In this example, the system was populated with data and APIs. These should first be tested to ensure that all published APIs respond as expected. Once this is done, use vSphere UI to complete the following steps:
 1. Power off VMs

To power off the VMs in sync, schedule a “shutdown guest OS” such that all the VMs shutdown at the same time. Although the shutdown does not occur at exactly the same instant, this does allow the shutdown for every machine to be as close in time as possible.
 2. Take snapshots

Take snapshots of each VM after shutdown. Using the scheduling feature in the UI, schedule a snapshot of all the VMs at the same instant. If cloning should adversely affect the system in any way then the original VMs can be reverted back to these snapshots.
 3. Clone VMs

Use the clone feature of the UI to create a clone of each VM. This VM should sit in the same resource pool as the original cluster. This is simply an exact copy of the disks of the original machine. The clones and originals are in no way coupled.
 4. Power on clones

Schedule the power-on for each clone VM such that all the clone VMs power on simultaneously. Once this is done, test the cluster to ensure that the APIs respond as expected. For the example cluster topology shown earlier on this page, it can take a couple of hours after power-on for all the APIs to respond as expected.

Disabling and re-enabling the Analytics subsystem on VMware

Disable the Analytics subsystem by shutting down the VM that hosts it.

Before you begin

This operation requires the use of the Analytics CLI. [Using the Analytics CLI](#).

About this task

Disabling the Analytics subsystem stops the collection and storage of data as well as making the subsystem unavailable.

When you disable Analytics, there will be no data in either the API Manager Dashboards or third-party offloads.

Procedure

1. Back up your Analytics data as explained in [Backing up and restoring the Analytics database on VMware](#).
2. Disable shard allocation for storage by following these steps:

Disabling shard allocation is optional, but is recommended because it prevents new shards from being created for replication when a node is unavailable, and helps avoid corruption issues in full system restarts.

- a. Create a json file called `analytics_persistent_settings.json` and insert the contents:

```
{"persistent":{"cluster.routing.allocation.enable":"none"}}
```

- b. Run the following Analytics CLI command:

```
apic --mode analytics clustermgmt:putSettings --server <management server platform api endpoint> --analytics-service <analytics service name> --format json analytics_persistent_settings.json
```

where:

- `<management server platform api endpoint>` is the Management server platform api endpoint.
- `<analytics service name>` is the name of Analytics service as configured in the Cloud Manager.

The output from this command should be:

```
{"acknowledged":true,"persistent":{"cluster":{"routing":{"allocation":{"enable":"none"}}},"transient":{}}
```

3. Perform a synced flush of storage by running the following command:

Flushing the storage is optional, but is recommended because it helps to avoid losing the data that was not yet written to disk before you restart pods. This step flushes all current index operations synchronously and attempts to write everything that is in flux to disk.

```
apic --mode analytics clustermgmt:postFlush --server <management server platform api endpoint> --analytics-service <analytics service name> --format json
```

where:

- `<management server platform api endpoint>` is the Management server platform endpoint.
- `<analytics service name>` is the name of Analytics service as configured in the Cloud Manager.

When you flush storage, the response looks like the following example, with the *failed* count set to zero:

```
{
  "_shards": {
    "total": 16,
    "successful": 6,
    "failed": 0
  }
}
```

This operation occasionally fails, but it is safe to rerun it multiple times until it passes and the “failed” count is zero.

4. Shut down the Analytics VMs.

You can shut down the VMs using one of the following methods:

- Use the VMware console
- `ssh` into the images and run the `shutdown` command

Re-enabling analytics

5. Start your analytics VMs.

6. Enable shard allocation by following these steps:

Note: If you did not disable shard allocation, skip this step.

a. Create a json file called `analytics_persistent_settings.json` and insert the contents:

```
{"persistent": {"cluster.routing.allocation.enable": "all"}}
```

b. Run the following Analytics CLI command:

```
apic --mode analytics clustermgmt:putSettings --server <management server platform api endpoint> --analytics-service <analytics service name> --format json analytics_persistent_settings.json
```

where:

- `<management server platform api endpoint>` is the Management server platform api endpoint.
- `<analytics service name>` is the name of Analytics service as configured in the Cloud Manager.

When you enable shard allocation, the response looks like the following example:

```
{"acknowledged": true, "persistent": {"cluster": {"routing": {"allocation": {"enable": "all"}}}}, "transient": {}}
```

Maintaining a two data center deployment

How to maintain a two data center disaster recovery (DR) deployment on VMware, including information about normal operation, failure handling, and recovery.

Ensure that you understand the concepts of a 2DCDR deployment in API Connect. For more information, see [A two data center deployment strategy on VMware](#).

See the following topics for information about failure handling, backup, upgrade, and removing a 2DCDR deployment.

- [Backup and restore considerations for a two data center deployment](#)
Points to consider when backing up and restoring a two data center disaster recovery (2DCDR) deployment on VMware.
- [Failure handling of a two data center deployment](#)
How to detect failures, and perform service failover, in a two data center disaster recovery deployment on VMware.
- [Recovering from a failover of a two data center deployment](#)
How to recover a two data center disaster recovery deployment on VMware.
- [Removing a two data center deployment](#)
You can remove your two data center disaster recovery deployment on VMware.

Related concepts

- [A two data center deployment strategy on VMware](#)

Related tasks

- [Installing a two data center deployment on VMware](#)

Backup and restore considerations for a two data center deployment

Points to consider when backing up and restoring a two data center disaster recovery (2DCDR) deployment on VMware.

The 2DCDR backup requirements are different for the Portal and Management subsystems.

Management subsystem 2DCDR backup requirements

- Different remote backup locations must be specified for each data center.
- Before you configure S3 backup settings you must first disable 2DCDR on both sites, and then re-enable 2DCDR after reconfiguring your backup settings.
 1. Disable 2DCDR for the management subsystems on both sites:

```
apicup subsys set <management subsystem> multi-site-ha-enabled=false
apicup subsys install <management subsystem>
```

2. Configure your S3 backup settings.

3. Enable 2DCDR for the management subsystems on both sites:

```
apicup subsys set <management subsystem> multi-site-ha-enabled=true
apicup subsys install <management subsystem>
```

You do not need to do this for SFTP or local backups.

- Both active and warm-standby Management subsystems run the scheduled backup job.
- Restores must be run on the active data center.

Portal subsystem 2DCDR backup requirements

- The backup settings for both the active and warm-standby portal must be identical.

- Only the active data center portal takes backups. On failover, the backup job of the new warm-standby is disabled, and the backup job of the new active is enabled.
- Portal restores are only done on the active data center.

Configuring backups for 2DCDR deployments

Keeping the different Portal and Management 2DCDR backup requirements in mind, the actual steps for configuring backups for both subsystems are the same as for a single API Connect deployment:

- [Backing up the management subsystem](#)
- [Restoring the management subsystem](#)
- [Backing up and restoring the Developer Portal](#)

Related concepts

- [A two data center deployment strategy on VMware](#)

Related tasks

- [Installing a two data center deployment on VMware](#)

Failure handling of a two data center deployment

How to detect failures, and perform service failover, in a two data center disaster recovery deployment on VMware.

Before you begin

Ensure that you understand the concepts of a 2DCDR deployment in API Connect. For more information, see [A two data center deployment strategy on VMware](#).

About this task

Failure handling comprises of two main concepts; failure detection, and failover of services.

Failure detection is about detecting that a failure has happened, and then alerting the relevant administrator for them to take action. Failure detection is different from system monitoring. Failure detection checks are run often and fast (for example, every 5 seconds). As such they are normally more limited in what they check. In API Connect, failure checks detect only whether the database and web server are online, and nothing more. Whereas monitoring checks typically are run less frequently, but are more likely to check for things like CPU, memory, and disk space usage. The results of these checks can then be recorded to perform historical trend analysis to spot memory leaks for example.

If a failure is detected, it's then important to make sure that the cluster or service is definitely offline. So, for example, if data center one (DC1) appears to have an outage, and the plan is to failover to data center two (DC2), then it's important to make sure that the pods in DC1 really are offline. If the DC1 pods are actually still running, then they might continue to send data to DC2, which would lead DC2 to think it wasn't the active deployment, and then that could lead to split-brain. A fast way to ensure that the cluster or service really is offline, is to drop the network link to those pods in your VMware infrastructure, and then delete them.

The failure detection endpoint must be location specific and must be exempt from any traffic routing, so that it's possible to explicitly check the health of each data center regardless of how traffic is being routed.

You can failover a specific service, for example `portal_service_1`, or all of the services that are running in a particular data center. The following sections describe how failover is achieved.

The operational states of an API Manager service:

Operational state	Description
<code>progressing to active</code>	Pods are progressing to the active state, but none are capable of serving traffic.
<code>active</code>	All of the pods are ready and in the correct disaster recovery state for the active data center.
<code>progressing to passive</code>	Pods are progressing to the warm-standby state, but none are capable of serving traffic.
<code>passive</code>	All of the pods are ready and in the correct disaster recovery state for the warm-standby data center.

The operational states of a Developer Portal service:

Operational state	Description
<code>progressing to active</code>	Pods are progressing to the active state, but none are capable of serving traffic.
<code>progressing to active (ready for traffic)</code>	At least one pod of each type is ready for traffic. The dynamic router can be linked to this service.
<code>active</code>	All of the pods are ready and in the correct disaster recovery state for the active data center.
<code>progressing to passive</code>	Pods are progressing to the warm-standby state, but none are capable of serving traffic.
<code>progressing to passive (ready for traffic)</code>	At least one pod of each type is ready for traffic.
<code>passive</code>	All of the pods are ready and in the correct disaster recovery state for the warm-standby data center.
<code>progressing to down</code>	Pods are moving from warm-standby state to down.
<code>down</code>	The multi-site HA mode is deleted from the warm-standby data center.

You can run the following command to check the operational state of a service:

```
kubect1 describe ServiceName
```

Where `ServiceName` is the name of the API Manager or Developer Portal service.

Important:

- If the active data center is offline, do not change the `multi-site-ha-enabled` setting to `false` on the warm-standby data center, as this action deletes all the data from the databases. If you want to revert to a single data center topology, you must change the `multi-site-ha-enabled` setting to `false` on the active

data center, and then redeploy the warm-standby data center. If the active data center is offline, you must first change the warm-standby data center to be active, and then change the `multi-site-ha-enabled` setting to `false` on the now active data center.

- If the warm-standby data center is offline for more than 24 hours, there can be issues with the disk space on the active data center so you must revert your deployment to a single data center topology. To revert to a single data center topology, you must change the `multi-site-ha-enabled` setting to `false` on the active data center. When the warm-standby site has been redeployed, the `multi-site-ha-enabled` setting can be reapplied to the active site.

Procedure

• API Manager service failover

To initiate a failover from DC1 to DC2 you first must set DC1 to warm-standby, before making the warm-standby data center active. This action is needed to prevent split-brain, as there cannot be two active services or clusters at the same time. If the failover is being done because the current active data center has had a complete outage, then you should ensure that the network links to that data center are dropped in your VMware infrastructure, in order to avoid the potential for split-brain occurring. The offline data center must be set to warm-standby before the network links are restored. The following instructions show how to failover DC1 to DC2 for the API Manager service.

1. Set DC1 (in this example, located in Dallas) to be warm-standby.

Run `apicup` to change the `multi-site-ha-mode` property to `passive` on the Dallas data center, for example:

```
apicup subsys set mgmt_dallas multi-site-ha-mode=passive
```

where

- `mgmt_dallas` is the name of the management service on DC1.

Note: If DC1 is completely down, and you cannot set it to be warm-standby at this point, you must ensure that the network links to DC1 are removed before continuing to set DC2 to be active. You must then not restore the network links to DC1 until you can set DC1 to be warm-standby.

2. Run `apicup` to update the settings on DC1, for example:

```
apicup subsys install mgmt_dallas
```

3. Log in to the virtual machine management subsystem on DC1 by using an SSH tool, and run the following command to check that the API Manager service is ready for traffic:

```
kubectl describe ServiceName
```

The service is ready for traffic when the `Ha mode` part of the `Status` object is set to `passive`. For example:

```
Status:
...
Ha mode:           passive
...
```

4. Set DC2 (in this example, located in Raleigh) to be active.

Run `apicup` to change the `multi-site-ha-mode` property to `active` on the Raleigh data center, for example:

```
apicup subsys set mgmt_raleigh multi-site-ha-mode=active
```

where

- `mgmt_raleigh` is the name of the management service on DC2.

5. Run `apicup` to update the settings on the services on DC2.

For example:

```
apicup subsys install mgmt_raleigh
```

6. Update your dynamic router to redirect all traffic to DC2 instead of DC1. Note that until the DC2 site becomes `active`, the UIs might not be operational.

7. Log in to the virtual machine management subsystem on DC2 by using an SSH tool, and run the following command to check that the API Manager service is ready for traffic:

```
kubectl describe ServiceName
```

The service is ready for traffic when the `Ha mode` part of the `Status` object is set to `active`. For example:

```
Status:
...
Ha mode:           active
...
```

• Developer Portal service failover

To initiate a failover for the Developer Portal from DC1 to DC2 you first must ensure that the existing DC1 is warm-standby, before making the warm-standby data center active, in order to prevent split-brain. The following instructions show how to failover DC1 to DC2 for the Developer Portal service. You must repeat these steps for each Developer Portal service that you want to failover.

Note:

- Switching a Developer Portal service to warm-standby mode instantly restarts all of its `portal-db` and `portal-www` pods at the same time. Therefore, if you don't have an active Developer Portal service that is online, there will be an outage of the Developer Portal web sites until the warm-standby data center is ready to accept traffic.
- If you want to monitor the failover process, you can run the following command to check the status of the Portal pods in the `PortalService` CR file:

```
kubectl describe ServiceName
```

For more details about the status information in a `PortalCluster` CR file, see the [Example](#) section.

- Do not edit the `PortalService` CR file during the failover process.

1. Set DC1 (in this example, located in Dallas) to be warm-standby.

Run `apicup` to change the `multi-site-ha-mode` property to `passive` on the Dallas data center, for example:

```
apicup subsys set port_dallas multi-site-ha-mode=passive
```

where

- `port_dallas` is the name of the Portal service on DC1.

Note: If DC1 is completely down, and you cannot set it to be warm-standby at this point, you must ensure that the network links to DC1 are removed before continuing to set DC2 to be active. You must then not restore the network links to DC1 until you can set DC1 to be warm-standby.

2. Run `apicup` to update the settings on DC1, for example:

```
apicup subsys install port_dallas --skip-health-check
```

Important: You must not wait for the active DC `status` to become `warm-standby`, before failing over the warm-standby DC to become `active`. You should change the DC1 `multi-site-ha-mode` to `passive`, and then run the `kubectl describe ServiceName` command, and as soon as the `haMode` is set to `progressing to passive`, you must change DC2 to be `active`.

3. Set DC2 (in this example, located in Raleigh) to be active.

Run `apicup` to change the `multi-site-ha-mode` property to `active` on the Raleigh data center, for example:

```
apicup subsys set port_raleigh multi-site-ha-mode=active
```

where

- `port_raleigh` is the name of the Portal service on DC2.

4. Run `apicup` to update the settings on the services on DC2.

For example:

```
apicup subsys install port_raleigh --skip-health-check
```

5. Update your dynamic router to redirect all traffic to DC2 instead of DC1.

6. Log in to the virtual machine management subsystem on DC2 by using an SSH tool, and run the following command to check that the Developer Portal service is ready for traffic:

```
kubectl describe ServiceName
```

The service is ready for traffic when the `haMode` part of the `Status` object is set to `active`. For example:

```
Status:
...
haMode: active
...
```

- **Entire data center failover**

To failover an entire datacenter, follow the previous steps to failover the API Manager service, followed by the steps to failover each Developer Portal service in that data center.

How long it takes to complete the failover varies, and depends on hardware speed, network latency, and the size of the databases. However, here are some approximate timings:

For API Manager:

- `warm-standby` to `active` approximately 5 minutes
- `active` to `warm-standby` approximately 15 minutes

For Developer Portal:

- `warm-standby` to `active` 15 - 40 minutes
- `active` to `warm-standby` approximately 10 minutes

Example

Example of the status section of a `PortalCluster` CR file for an active data center that you can view by running the `kubectl describe PortalService` command:

```
status:
  conditions:
  - lastTransitionTime: "2020-04-27T08:00:25Z"
    message: 4/6
    status: "False"
    type: Ready
  customImages: true
  dbCASecret: portal-db-ca
  encryptionSecret: ptl-encryption-key
  endpoints:
    portal-director: https://api.portal.ha-demo.cloud/
    portal-web: https://portal.ha-demo.cloud/
    replication: https://ptl-replication.cvs.apic-2ha-dallas-144307-xxxx14539d6e75f74e950db7077951d4-0000.us-
south.containers.appdomain.cloud/
  localMultiSiteHAPeerConfigHash: UM9VtuWDDp/328n0oCl/GQGFzy34o1V9sH2OyXxKMHw=
  haMode: progressing to active (ready for traffic)
  microServiceSecurity: custom
  multiSiteHA:
    dbPodScale: 3
    remoteSiteDeploymentName: portal
    remoteSiteName: frankfurt
  replicationPeer: https://ptl-replication.cvs.apic-2ha-frankfur-683198-xxxx14539d6e75f74e950db7077951d4-0000.eu-
de.containers.appdomain.cloud/
  wwwPodScale: 3
  phase: 0
  serviceCASecret: portal-ca
  serviceClientSecret: portal-client
  serviceServerSecret: portal-server
  services:
    db: portal-dallas-db
    db-remote: portal-frankfurt-db
    nginx: portal-nginx
    tunnel: portal-tunnel
    www: portal-dallas-www
    www-remote: portal-frankfurt-www
  versions:
```

```

available:
  channels:
    - name: "10"
    - name: "10.0"
    - name: 10.0.0
    - name: 10.0.0.0
  versions:
    - name: 10.0.0.0-1038
reconciled: 10.0.0.0-1038

```

Meaning of the **haMode** for the active data center:

- **active** - this is the active data center, or the only data center, and all pods are ready and accepting traffic.
- **progressing to active** - this is the active data center, or the only data center, and there are no pods ready yet.
- **progressing to active (ready for traffic)** - this is the active data center, or the only data center, and there is at least one pod of each type, www, db, and nginx, ready, so the data center is accepting traffic.

Example of the status section of a **PortalCluster** CR file for a warm-standby data center that you can view by running the **kubectl describe PortalService** command:

```

status:
  conditions:
    - lastTransitionTime: "2020-04-21T10:54:54Z"
      message: 6/6
      status: "True"
      type: Ready
  customImages: true
  dbCASecret: portal-db-ca
  encryptionSecret: ptl-encryption-key
  endpoints:
    portal-director: https://api.portal.ha-demo.cloud/
    portal-web: https://portal.ha-demo.cloud/
    replication: https://ptl-replication.cvs.apic-2ha-frankfur-683198-xxxx14539d6e75f74e950db7077951d4-0000.eu-
de.containers.appdomain.cloud/
  localMultiSiteHAPeerConfigHash: VguODE74TkiS3Lcc5ytQiaF8100PXMHUrVBtb+PbKOG=
  haMode: progressing to passive (ready for traffic)
  microServiceSecurity: custom
  multiSiteHA:
    dbPodScale: 3
    remoteSiteDeploymentName: portal
    remoteSiteName: dallas
    replicationPeer: https://ptl-replication.cvs.apic-2ha-dallas-144307-xxxx14539d6e75f74e950db7077951d4-0000.us-
south.containers.appdomain.cloud/
    wwwPodScale: 3
  phase: 0
  serviceCASecret: portal-ca
  serviceClientSecret: portal-client
  serviceServerSecret: portal-server
  services:
    db: portal-frankfurt-db
    db-remote: portal-dallas-db
    nginx: portal-nginx
    tunnel: portal-tunnel
    www: portal-frankfurt-www
    www-remote: portal-dallas-www
  versions:
    available:
      channels:
        - name: "10"
        - name: "10.0"
        - name: 10.0.0
        - name: 10.0.0.0
      versions:
        - name: 10.0.0.0-1038
    reconciled: 10.0.0.0-1038

```

Meaning of the **haMode** for the warm-standby data center:

- **passive** - this is the warm-standby data center, and all pods are ready and accepting traffic.
- **progressing to passive** - this is the warm-standby data center, and there are no pods ready yet.
- **progressing to passive (ready for traffic)** - this is the warm-standby data center, and there is at least one pod of each type, www, db, and nginx, ready, so the data center is accepting traffic.

What to do next

As soon as a failure on a data center has been resolved, the failed data center should be brought back online and re-linked to the currently active data center in order to maintain the highest availability; see [Recovering from a failover of a two data center deployment](#) for more information.

Related concepts

- [A two data center deployment strategy on VMware](#)

Related tasks

- [Recovering from a failover of a two data center deployment](#)
- [Installing a two data center deployment on VMware](#)

Recovering from a failover of a two data center deployment

How to recover a two data center disaster recovery deployment on VMware.

Before you begin

Ensure that you understand the concepts of a 2DCDR deployment in API Connect. For more information, see [A two data center deployment strategy on VMware](#).

About this task

After a failure has been resolved, the failed data center can be brought back online and re-linked to the currently active data center. It's important to do this as soon as possible, in order to reinstate disaster recovery. Otherwise, if another failure occurred before the failed data center is brought back online, there could be a complete outage.

The failed data center can be brought back online as the new active primary data center, or it can be brought back as a warm-standby secondary data center and the currently active data center kept as the primary one. This decision will be based on your own company policies about recovering from a failure.

The amount of data that is sent when recovering from a failover, will depend on how long the outage lasted, and how much activity there was during that period.

Important:

- If the active data center is offline, do not change the `multi-site-ha-enabled` setting to `false` on the warm-standby data center, as this action deletes all the data from the databases. If you want to revert to a single data center topology, you must change the `multi-site-ha-enabled` setting to `false` on the active data center, and then redeploy the warm-standby data center. If the active data center is offline, you must first change the warm-standby data center to be active, and then change the `multi-site-ha-enabled` setting to `false` on the now active data center.
- If the warm-standby data center is offline for more than 24 hours, there can be issues with the disk space on the active data center so you must revert your deployment to a single data center topology. To revert to a single data center topology, you must change the `multi-site-ha-enabled` setting to `false` on the active data center. When the warm-standby site has been redeployed, the `multi-site-ha-enabled` setting can be reapplied to the active site.

Procedure

• Recovering from an API Manager failover

The following instructions show how to bring data center one (DC1) back as the primary (**active**) data center, and assume that DC1 is now online and in the warm-standby state. If you prefer, you can keep DC1 as the secondary data center, and leave data center two (DC2) as the primary (**active**) data center.

1. Set DC2 (in this example, located in Raleigh) to be warm-standby.

Run `apicup` to change the `multi-site-ha-mode` property to `passive` on the Raleigh data center, for example:

```
apicup subsys set mgmt_raleigh multi-site-ha-mode=passive
```

where

- `mgmt_raleigh` is the name of the management service on DC2.

2. Run `apicup` to update the settings on DC2, for example:

```
apicup subsys install mgmt_raleigh
```

3. Log in to the virtual machine management subsystem on DC2 by using an SSH tool, and run the following command to check that the API Manager service is ready for traffic:

```
kubectl describe ServiceName
```

The service is ready for traffic when the `Ha mode` part of the `Status` object is set to `passive`. For example:

```
Status:
...
Ha mode:           passive
...
```

You must also ensure that the database in DC2 has replicated over completely before proceeding.

4. Set DC1 (in this example, located in Dallas) to be active.

Run `apicup` to change the `multi-site-ha-mode` property to `active` on the Dallas data center, for example:

```
apicup subsys set mgmt_dallas multi-site-ha-mode=active
```

where

- `mgmt_dallas` is the name of the management service on DC1.

5. Run `apicup` to update the settings on DC1.

For example:

```
apicup subsys install mgmt_dallas
```

6. Update the dynamic router to redirect all traffic to DC1 instead of DC2. Note that until the DC1 site becomes **active**, the UIs might not be operational.
7. Log in to the virtual machine management subsystem on DC1 by using an SSH tool, and run the following command to check that the API Manager service is ready for traffic:

```
kubectl describe ServiceName
```

The service is ready for traffic when the `Ha mode` part of the `Status` object is set to `active`. For example:

```
Status:
...
Ha mode:           active
...
```

- **Recovering from a Developer Portal failover**

When recovering from a Developer Portal failover, how much data is sent from the now warm-standby data center (in this example, DC2) back to the newly recovered active data center (in this example, DC1) varies depending on how much time has passed and how many changes have been made in the meantime.

If possible, the Portal sends an Incremental State Transfer (IST), which is a replay of the transactions that have happened since the failed data center, DC1, became unreachable. The IST can grow to be up to 3 GB in size. If the IST grows larger than 3 GB, then the Portal sends a State Snapshot Transfer (SST) instead, which is basically a full copy of the whole database. How long this process takes depends on how many Portal sites there are, how much content in each of those sites, how active they are (as in number of transactions), and how fast the network speed is between the data centers. However, it is likely to take in the range 5 - 60 minutes.

You can check whether the Developer Portal servers are in sync by running the `status` command in the `portal-www` pod admin container. If all servers show as `primary`, then they are in sync and traffic can be directed wherever needed. For information about running the status command, see [Verifying deployment of the Developer Portal subsystem](#).

Note:

- If you want to monitor the recovery process, you can log in to the virtual machine Portal subsystem by using an SSH tool, and run the following command to check the status of the Portal pods in the `PortalService` CR file:

```
kubectl describe ServiceName
```

For more details about the status information in a `PortalCluster` CR file, see the [Example](#) section.

- Do not edit the `PortalService` CR file during the recovery process.

The following instructions show how to bring DC1 back as the active primary data center, and switch DC2 to be the warm-standby secondary data center. Again, you can keep DC2 as the active primary data center if you prefer, and leave DC1 to be warm-standby.

1. Set DC2 (in this example, located in Raleigh) to be warm-standby.
Run `apicup` to change the `multi-site-ha-mode` property to `passive` on the Raleigh data center, for example:

```
apicup subsys set port_raleigh multi-site-ha-mode=passive
```

where

- `port_raleigh` is the name of the Portal service on DC2.

2. Run `apicup` to update the settings on the services on DC2.
For example:

```
apicup subsys install port_raleigh --skip-health-check
```

Important: You must not wait for the active DC `Status` to become `warm-standby`, before changing the warm-standby DC to become `active`. You should change the DC2 `multi-site-ha-mode` to `passive`, and then run the `kubectl describe ServiceName` command, and as soon as the `haMode` is set to `progressing to passive`, you must change DC1 to be `active`.

3. Set DC1 (in this example, located in Dallas) to be active.
Run `apicup` to change the `multi-site-ha-mode` property to `active` on the Dallas data center, for example:

```
apicup subsys set port_dallas multi-site-ha-mode=active
```

where

- `port_dallas` is the name of the Portal service on DC1.

4. Run `apicup` to update the settings on the services on DC1.
For example:

```
apicup subsys install port_dallas --skip-health-check
```

5. Update the dynamic router to redirect all traffic to DC1 instead of DC2.
6. Log in to the virtual machine Portal subsystem on DC1 by using an SSH tool, and run the following command to check that the Developer Portal service is ready for traffic:

```
kubectl describe ServiceName
```

The service is ready for traffic when the `Ha mode` part of the `Status` object is set to `active`. For example:

```
Status:
...
Ha mode: active
...
```

Related concepts

- [A two data center deployment strategy on VMware](#)

Related tasks

- [Failure handling of a two data center deployment](#)
- [Installing a two data center deployment on VMware](#)

Removing a two data center deployment

You can remove your two data center disaster recovery deployment on VMware.

Before you begin

Ensure that you understand the concepts of a 2DCDR deployment in API Connect. For more information, see [A two data center deployment strategy on VMware](#).

About this task

If you decide to remove your two data center disaster recovery (DR) deployment, you go from two data centers to one. The active data center remains as the only working system. The warm-standby one loses all data as part of the process.

Warning: If the active data center is offline, do not remove the multi-site HA CR section from the warm-standby data center, as this action deletes all the data from the databases.

As part of the installation, you configured the `multi-site-ha` settings on each data center, as described in the following table:

Table 1. The `multi-site-ha` settings

Setting	Description
<code>multi-site-ha-enabled</code>	Indicates whether two data center DR deployment is enabled. Set to <code>true</code> to enable two data center DR. Set to <code>false</code> to disable two data center DR. The default value is <code>false</code> .
<code>multi-site-ha-mode</code>	The state of the data center in the context of the two data center deployment. Valid values are: <ul style="list-style-type: none"><code>active</code> - indicates the primary data center<code>passive</code> - indicates the warm-standby data center
<code>management-replication</code>	Required only for the Management subsystem. The external ingress name for the Management subsystem in the current data center in the two data center deployment. The name is a unique fully qualified hostname that the other data center uses to communicate with the current data center.
<code>portal-replication</code>	Required only for the Portal subsystem. The external ingress name for the Portal subsystem in the current data center in the two data center deployment. The name is a unique fully qualified hostname that the other data center uses to communicate with the current data center.
<code>replication-peer-fqdn</code>	The ingress hostname for the other data center in the two data center deployment. This information is required so that the two data centers can communicate with each other.

If you want to revert to a single data center topology, or temporarily remove the warm-standby data center from DR, you must disable the `multi-site-ha` configuration on the active data center, and then remove the warm-standby data center. If the active data center is offline, you must first change the warm-standby data center to be active, and then disable the `multi-site-ha` configuration on the now active data center.

Procedure

- Disable the `multi-site-ha` configuration on the Management subsystem of the active data center. For example:

```
apicup subsys set mgmt multi-site-ha-enabled=false
apicup subsys install mgmt
```

- Disable the `multi-site-ha` configuration on the Portal subsystem of the active data center. For example:

```
apicup subsys set ptl multi-site-ha-enabled=false
apicup subsys install ptl
```

- You can now power off and shut down your warm-standby data center.

Related concepts

- [Maintaining a two data center deployment](#)
- [Backup and restore considerations for a two data center deployment](#)

Related tasks

- [Failure handling of a two data center deployment](#)
- [Recovering from a failover of a two data center deployment](#)

Monitoring and health checks

Monitor your deployment and complete health checks on subsystems.

See:

- [Checking cluster health on VMware](#)
You can use `apicup` to check the health of the API Connect clusters in your VMware deployment.
- [Monitoring Postgres disk usage on VMware](#)
Monitor disk usage by the Postgres database in the management subsystem, when deployed on VMware.
- [Obtaining simple health check data of Developer Portal sites by using a REST API call](#)
Call a simple health check API from your external load balancer to dynamically determine whether a specific Developer Portal site in a cluster is working. This API call can be used by a load balancer to help determine where to route traffic.
- [Monitoring the network with SNMP](#)
Presents a table of OID trees that you can poll using SNMP Get.

Checking cluster health on VMware

You can use `apicup` to check the health of the API Connect clusters in your VMware deployment.

The `health-check` command checks a number of criteria to determine the health of their cluster. When all criteria are successfully met, the command displays no output, and exits with a status of 0. When one or more criteria are not met, the command stops processing and displays a message with the failure and exits with a status

of 1.

The command takes no arguments. The only option is the `--verbose` flag. If `--verbose` is set the command prints all checks that were performed.

The `health-check` is run against the namespace that is specified in `apiconnect-up.yaml`.

The health check is run on the specified subsystem.

The command verifies that for the specified subsystem:

- The `apicup` version matches the API Connect release version.
- All cluster members are running the same API Connect release version.
- All cluster members have a deployment status of `Done`.
- Docker is running.
- `kubelet` is running
- All Kubernetes-defined nodes are running.
- The Kubernetes Control Plane pods are running:
 - `etcd/kube-apiserver`
 - `kube-system.kube-controller`
 - `kube-scheduler`
 - `kube-apiserver-proxy`
- The add-on Kubernetes Deployments are fulfilled:
 - `coredns`
 - `metrics-server`
- The add-on Kubernetes DaemonSets are fulfilled:
 - `calico-node`
 - `ingress-nginx-ingress-controller`
 - `kube-proxy`

Syntax:

Usage:

```
apicup subsys health-check <SUBSYS> [flags]
```

Flags:

```
-h, --help                help for health-check
--kubeconfig string       (optional) absolute path to the kubeconfig file (default "/Users/<username>/.kube/config")
-v, --verbose             Verbose output
```

Global Flags:

```
--accept-license  Accept the license for API Connect
--debug           Enable debug logging
```

Example usage, for a subsystem named `mgmt`:

```
../apicup subsys health-check mgmt
```

Note:

- The command flag `--kubeconfig string` does not apply to deployments on VMware.
- In a multi-node OVA cluster, the `apicup subsys health-check` command does not return status information if a majority of the nodes are down. To obtain status information, start more nodes. Status is returned only when a quorum is achieved.

Monitoring Postgres disk usage on VMware

Monitor disk usage by the Postgres database in the management subsystem, when deployed on VMware.

Postgres Database is the core database used in Management Subsystem. It is important to monitor the Postgres disk usage.

In v10.0.3.0 and greater, the APICoconnect operator tracks the current disk usage of the Postgres components, and regularly updates `apic health-check` command.

APICoconnect operator regularly updates the `ManagementCluster` instance status which is reflected in the `apic health-check` command.

`apic health-check` command reports WARNING state when disk utilization reaches 50%.

`apic health-check` command reports ERROR state when disk utilization reaches 70% and Postgres is brought down.

See:

- [Management Cluster disk stats](#)
- [Warning condition is populated at 50% usage](#)
- [Error condition is populated at 70% usage](#)

Management Cluster disk stats

The `ManagementCluster` instance has `.status.postgresDataStats` where the operator prints the current disk usage of Postgres components. Use `kubectl` to view the status.

To use `kubectl`, first ssh into the appliance as `root`.

```
kubectl get mgmt ml -o json | jq .status.postgresDataStats
[
  {
```

```

"instanceName": "m1-ed60c42d-postgres",
"podName": "m1-ed60c42d-postgres-86766f69cb-xs7t5",
"pvcCapacity": 250181844992,
"pvcName": "m1-ed60c42d-postgres",
"pvcType": "PostgreSQL",
"pvcUsed": 60895232,
"pvcUsedPercentage": 0
},
{
"instanceName": "m1-ed60c42d-postgres-backrest-shared-repo",
"podName": "m1-ed60c42d-postgres-backrest-shared-repo-859484b5f6-r7cv6",
"pvcCapacity": 250181844992,
"pvcName": "m1-ed60c42d-postgres-pgbr-repo",
"pvcType": "pgBackRest",
"pvcUsed": 16203776,
"pvcUsedPercentage": 0
},
{
"instanceName": "m1-ed60c42d-postgres",
"podName": "m1-ed60c42d-postgres-86766f69cb-xs7t5",
"pvcCapacity": 250181844992,
"pvcName": "m1-ed60c42d-postgres-wal",
"pvcType": "WAL",
"pvcUsed": 201338880,
"pvcUsedPercentage": 0
},
{
"instanceName": "m1-ed60c42d-postgres-fqbg",
"podName": "m1-ed60c42d-postgres-fqbg-84869d976c-49ggc",
"pvcCapacity": 250181844992,
"pvcName": "m1-ed60c42d-postgres-fqbg",
"pvcType": "PostgreSQL",
"pvcUsed": 59994112,
"pvcUsedPercentage": 0
},
{
"instanceName": "m1-ed60c42d-postgres-fqbg",
"podName": "m1-ed60c42d-postgres-fqbg-84869d976c-49ggc",
"pvcCapacity": 250181844992,
"pvcName": "m1-ed60c42d-postgres-fqbg-wal",
"pvcType": "WAL",
"pvcUsed": 201334784,
"pvcUsedPercentage": 0
},
{
"instanceName": "m1-ed60c42d-postgres-cimo",
"podName": "m1-ed60c42d-postgres-cimo-5769868b75-5z7ln",
"pvcCapacity": 250181844992,
"pvcName": "m1-ed60c42d-postgres-cimo",
"pvcType": "PostgreSQL",
"pvcUsed": 59994112,
"pvcUsedPercentage": 0
},
{
"instanceName": "m1-ed60c42d-postgres-cimo",
"podName": "m1-ed60c42d-postgres-cimo-5769868b75-5z7ln",
"pvcCapacity": 250181844992,
"pvcName": "m1-ed60c42d-postgres-cimo-wal",
"pvcType": "WAL",
"pvcUsed": 201334784,
"pvcUsedPercentage": 0
}
}
]

```

Warning condition is populated at 50% usage

When one of the Postgres components occupy 50% usage, the operator marks the overall status of the **ManagementCluster** to **Warning** with an appropriate warning message, which is included in the output from **apic health-check** command.

The warning condition explains why the warning condition was set.

To use **apic**, first ssh into the appliance as **root**.

```
apic health-check
```

```
FATA[0007] Cluster not in good health:
ManagementCluster is not Ready or Complete | State: 17/17 Phase:
Warning Message: Current WAL disk usage of nihar-stac-3955b42d-sitel-postgres is 51 percent.
DATABASE SHUTDOWN starts at 70 percent utilization. Please contact IBM Support immediately
```

If you encounter the warning, contact IBM support to fix the root cause.

Error condition is populated at 70% usage

An error condition is populated when the operator brings down Postgres because it is using 70% of available disk space. Postgres is brought down to avoid problems, such as disk corruption, that can occur if the disk becomes completely filled.

To use **apic**, first ssh into the appliance as **root**.

```
apic health-check
```

```
FATA[0011] Cluster not in good health:
ManagementCluster is not Ready or Complete | State: 17/17 Phase:
Error Message: Current WAL disk usage of nihar-stac-3955b42d-site1-postgres is more than 70 percent,
initiating DATABASE SHUTDOWN
```

If you encounter the error, contact IBM support to fix the root cause.

Important:

On VMware, because the entire disk is allocated to the worker node, in some cases before the operator reacts to 70% usage, , Kubernetes itself can face disk pressure and starts evicting pods. In these cases, increase the disk allocated to worker node in order to stabilize the worker node. See [Adding disk space to a VMware appliance](#).

Obtaining simple health check data of Developer Portal sites by using a REST API call

Call a simple health check API from your external load balancer to dynamically determine whether a specific Developer Portal site in a cluster is working. This API call can be used by a load balancer to help determine where to route traffic.

About this task

You can use the site health REST API to determine whether a particular Developer Portal site is running. The site health API returns the current system time of the site if both the database and web server are running. This API is fast and puts no load on the system, so it is ideal for use with load balancers to help them determine where to route traffic.

Procedure

To call the site health REST API, append `/health` to the end of your Developer Portal site URL in your web browser, as follows:

```
site_url/health
```

Where `site_url` is the URL of the Developer Portal site that you want to check.

If both the database and web server of the site are running, the web browser returns the current system time. For example:

```
1511367695
```

If either the database or the web server of the site is not running, the web browser returns an error that the site can't be reached.

If you do get an error from the `/health` endpoint, refer to the [must gather logs page](#) and obtain the `portal-www` web logs, so that you can see more information on the error.

Monitoring the network with SNMP

Presents a table of OID trees that you can poll using SNMP Get.

Simple Network Management Protocol (SNMP) collects information from network devices, such as servers, printers, hubs, switches, and routers on an Internet Protocol (IP). SNMP employs addresses for network devices using a hierarchical numbering system called Object Identifiers (OID). The following table provides OID addresses for polling in SNMP to monitor API Connect servers.

Note:

- Only SNMP v2 and SNMP v3 are supported. However, it is not recommended to enable SNMP v2. Instead, configure SNMPv3 as shown in [Configuring SNMPv3](#), and then enable it as shown in [Enabling SNMP](#).
- If the server has two or more network interfaces, the SNMP request is sent only to the first network interface.
- SNMP is disabled by default for all hosts.
- This information applies to API Connect deployments in VMware environments. It does not apply to Kubernetes deployments.

Each entry in the table represents many individual items. Use `snmpwalk` or another SNMP polling utility to see the complete list. You can poll the following OID trees:

Table 1. Default SNMPv2 configuration

OID	SNMP Name	Notes
.1.3.6.1.2.1.1	SNMPv2-MIB::system	
.1.3.6.1.2.1.2	IF-MIB::interfaces	
.1.3.6.1.2.1.4	IP-MIB::ip	
.1.3.6.1.2.1.5	IP-MIB::icmp.	
.1.3.6.1.2.1.6	TCP-MIB::tcp	
.1.3.6.1.2.1.7	UDP-MIB::udp	
.1.3.6.1.2.1.11	SNMPv2-MIB::snmp	
.1.3.6.1.2.1.25.1	HOST-RESOURCES-MIB::hrSystem	Excluding .1.3.6.1.2.1.25.1.3 HOST-RESOURCES-MIB::hrSystemInitialLoadDevice Excluding .1.3.6.1.2.1.25.1.4 HOST-RESOURCES-MIB::hrSystemInitialLoadParameters
.1.3.6.1.2.1.25.2	HOST-RESOURCES-MIB::hrStorage	
.1.3.6.1.2.1.25.3	HOST-RESOURCES-MIB::hrDevice	
.1.3.6.1.4.1.2021.4	UCD-SNMP-MIB::memory	
.1.3.6.1.4.1.2021.10	UCD-SNMP-MIB::laTable	CPU Load Average
.1.3.6.1.4.1.2021.11	UCD-SNMP-MIB::systemStats	

Configuring SNMPv3

From an OVA shell prompt as root edit the file `/etc/snmp/snmpd.conf` so that its content is:

```
createUser readOnlyUser SHA authpassphrase AES privpassphrase
rouser readOnlyUser
agentAddress udp:161,udp6:[::1]:161
```

- Replace `authpassphrase` with the passphrase to be used for authentication,
- Replace `privpassphrase` with the passphrase to be used for privacy (encryption) of the communications.

Enabling SNMP

It is not recommended to start SNMP without first configuring it for SNMPv3. See [Configuring SNMPv3](#).

Enable SNMP on each individual OVA node with the following commands from an OVA shell prompt as root:

```
systemctl start snmpd
systemctl enable snmpd
```

Sample output:

```
root@mgmt:~# systemctl start snmpd
root@mgmt:~# systemctl enable snmpd
snmpd.service is not a native service, redirecting to systemd-sysv-install
Executing /lib/systemd/systemd-sysv-install enable snmpd
inserv: warning: current start runlevel(s) (empty) of script `snmpd' overrides LSB defaults (2 3 4 5).
inserv: warning: current stop runlevel(s) (0 1 2 3 4 5 6) of script `snmpd' overrides LSB defaults (0 1 6).
```

Disabling SNMP

Disable SNMP on each individual OVA node with the following commands from an OVA shell prompt as root:

```
systemctl stop snmpd
systemctl disable snmpd
```

Sample output:

```
root@mgmt:~# systemctl stop snmpd
root@mgmt:~# systemctl disable snmpd
snmpd.service is not a native service, redirecting to systemd-sysv-install
Executing /lib/systemd/systemd-sysv-install disable snmpd
inserv: warning: current start runlevel(s) (empty) of script `snmpd' overrides LSB defaults (2 3 4 5).
inserv: warning: current stop runlevel(s) (0 1 2 3 4 5 6) of script `snmpd' overrides LSB defaults (0 1 6).
```

Checking SNMP status

Check whether SNMP is enabled on each individual OVA node with the following commands from an OVA shell prompt:

```
systemctl status snmpd
```

Sample output when SNMP is enabled and running:

```
apicadm@mgmt:~$ systemctl status snmpd
snmpd.service - LSB: SNMP agents
Loaded: loaded (/etc/init.d/snmpd; bad; vendor preset: enabled)
Active: active (running) since Wed 2019-02-06 03:53:02 UTC; 2 days ago
Docs: man:systemd-sysv-generator(8)
Tasks: 1
Memory: 6.0M
CPU: 3min 1.641s
CGroup: /system.slice/snmpd.service
└─1366 /usr/sbin/snmpd -Lsd -Lf /dev/null -u snmp -g snmp -I -smux mteTrigger mteTriggerConf -p /run/snmpd.pid
...
```

Sample output when SNMP is stopped and disabled:

```
root@mgmt:~# systemctl status snmpd
snmpd.service - LSB: SNMP agents
Loaded: loaded (/etc/init.d/snmpd; bad; vendor preset: enabled)
Active: inactive (dead)
Docs: man:systemd-sysv-generator(8)
...
```

Logging

Configure logging for your deployment on VMware.

See:

- [Configuring remote logging for a VMware deployment](#)
Logging collection is required for IBM Support to assist with troubleshooting. You can configure fluent-bit to collect and forward logs to a remote Syslog server.
- [Gathering logs for a VMware environment](#)
The `generate_postmortem.sh` script gathers all logs for troubleshooting and diagnostics.
- [Changing logging levels](#)
You can enable logging for entry and exit trace and for large payloads for apim-v2 pods.

Configuring remote logging for a VMware deployment

Logging collection is required for IBM Support to assist with troubleshooting. You can configure fluent-bit to collect and forward logs to a remote Syslog server.

Before you begin

Beginning in version 10.0.5.1, API Connect uses [fluent-bit](#) to collect log files for appliance-based deployments using OVA and forward the logs to a remote Syslog server.

In API Connect 10.0.5.2 or later, the fluent-bit service is already installed on the appliances and is configured to forward container logs to `/var/log/syslog` on the local machine. For best results, upgrade to API Connect 10.0.5.2 or later so you can take advantage of the included fluent-bit service.

Important: This article refers to third-party software that IBM does not control. As such, the software may change and this information might become outdated. In addition, the fluent-bit service was tested only for simple container log routing to local and remote Syslog servers as configured in the instructions that follow. Any customization of other fluent-bit settings (for example, to set up parsers or filters) is at your own risk and should be performed with caution by referring to the [fluent-bit documentation](#).

About this task

Remote logs created with fluent-bit use the following format:

```
Mar 1 00:38:44 apicdev1147 calico-node-gqbv_42428a8ade169ad3 2023-03-01 00:38:44.684 [INFO][208] felix/int_dataplane.go 1245: Applying dataplane updates
```

Where:

- `Mar 1 00:38:44` is the time the log arrived on the remote server.
- `apicdev1147` is the hostname of the log source.
- `calico-node-gqbv_42428a8ade169ad3` is a portion of the pod name/container_id where the log is coming from.
- `2023-03-01 00:38:44.684` is the timestamp at which the logged event actually took place within the container.
- The rest of the log is the log message itself.

Logs posted to your local server look different, as in this example:

```
Feb 28 23:47:03 apicdev1147 fluent-bit[2332273]: [2] kube.var.log.containers.kqn2-management-portal-proxy-647b96b89f-pkz5w_default_portal-proxy-74a1723512574fdc23fb9c014b80c7e83aff225b9d44d995736251edd18c49b4.log: [1677628023.503190040, {"_p">"F", "log">"Tue, 28 Feb 2023 23:47:03 GMT express:router expressInit : /healthz", "tag">"kqn2-management-74a1723512574fdc", "time">"2023-02-28T23:47:03.503190005Z", "stream">"stderr"}]
```

On the local log, the whole fluent-bit JSON object is logged, which provides slightly more information (for example, the full file name of the log file) than in the remote log.

To configure remote logging, use the appropriate instructions for your deployed version of API Connect:

- [API Connect 10.0.5.2 and later: Configuring remote logging](#)
- [API Connect 10.0.5.2 and later: Update the remote logging configuration](#)
- [API Connect 10.0.5.1: Installing and configuring remote logging](#)
- [API Connect 10.0.5.1: Update the remote logging configuration](#)

API Connect 10.0.5.2 and later: Configuring remote logging

About this task

The fluent-bit service is already installed on the appliance and is configured to forward container logs to `/var/log/syslog` on the local machine. If you want to change the output destination to a remote Syslog server, complete the following steps:

Procedure

1. Create the `/etc/fluent-bit/append-tag.lua` file.
 - a. Paste the following code into the new file:

```
function append_tag(tag, timestamp, record)
    new_record = record
    local pod_name = string.sub(string.match(tag, "^kube%.var%.log%.containers%. (.+)$"), 1, 16)
    local container_id = string.sub(string.match(tag, "-([^-]+)%.log$"), 1, 16)

    new_record["tag"] = pod_name .. "-" .. container_id
    return 1, timestamp, new_record
end
```

- b. Save and close the file.
2. Create the `/etc/fluent-bit/fluent-bit-override.conf` file.
 - a. Add the following statements to the new file:
Replace the variables in the `[OUTPUT]` section with values for your deployment:

```
[SERVICE]
    Daemon Off
    Flush 1
    Log_Level info
    Parsers_File parsers.conf
    HTTP_Server On
    HTTP_Listen 0.0.0.0
    HTTP_Port 2021
    Health_Check On
```

```

[INPUT]
  Name tail
  Path /var/log/containers/*.log
  multiline.parser docker, cri
  Tag kube.*
  Mem_Buf_Limit 5MB
  Skip_Long_Lines On

[INPUT]
  Name systemd
  Tag host.*
  Systemd_Filter _SYSTEMD_UNIT=kubelet.service
  Read_From_Tail On

[FILTER]
  Name kubernetes
  Match kube.*
  Merge_Log On
  Keep_Log Off
  K8S-Logging.Parser On
  K8S-Logging.Exclude On

[FILTER]
  Name lua
  Match *
  Script /etc/fluent-bit/append-tag.lua
  call append_tag

[OUTPUT]
  Name syslog
  Match *
  Host <host>
  port 514
  mode udp
  syslog_format rfc5424
  syslog_maxsize 2048
  syslog_hostname_key hostname
  syslog_hostname_preset <hostname of local machine/log source>
  syslog_apname_key tag
  syslog_message_key log

```

Note: The default values shown in the [OUTPUT] section can be modified as explained in the [fluent-bit documentation](#).

- b. Save and close the file.
3. Run the following command to restart the fluent-bit service so it picks up the configuration changes:

```
systemctl restart appliance-manager && systemctl restart fluent-bit
```

API Connect 10.0.5.2 and later: Update the remote logging configuration

To update the fluent-bit configuration, complete the following steps:

Procedure

1. Run the following command to edit the configuration file:
- ```
vim /etc/fluent-bit/fluent-bit-override.conf
```
2. Complete your configuration changes, then save and close the file.
  3. Run the following command to restart the appliance-manager:

```
systemctl restart appliance-manager
```

## API Connect 10.0.5.1: Installing and configuring remote logging

### About this task

Download Debian packages from IBM Fix Central to install fluent-bit by completing the following steps:

### Procedure

1. If you previously performed a Helm installation of fluent-bit, uninstall it by running the following command:
- ```
helm uninstall fluent-bit
```
2. Download the following files from IBM Fix Central:
 - [libpq5_12.13-0ubuntu0.20.04.1_amd64.deb](#)
 - [fluent-bit_2.0.6_amd64.deb](#)
 3. Install libpq5 by running the following command:

```
apt install <path-to-libpq5_12.13-0ubuntu0.20.04.1_amd64.deb>
```

4. Install fluent-bit by running the following command:
Be sure to install libpq5 first because it is a prerequisite for fluent-bit.

```
apt install <path-to-fluent-bit_2.0.6_amd64.deb>
```

5. Create the /etc/fluent-bit/append-tag.lua file.
 - a. Paste the following code into the new file:

```
function append_tag(tag, timestamp, record)
  new_record = record

```

```

local pod_name = string.sub(string.match(tag, "^kubernetes.var.log.containers%.(.+$)", 1, 16)
local container_id = string.sub(string.match(tag, "-([^-]+).log$"), 1, 16)

new_record["tag"] = pod_name .. "_" .. container_id
return 1, timestamp, new_record
end

```

b. Save and close the file.

6. Configure the input and output settings for the configuration file, so that fluent-bit forwards logs to Syslog:

a. Run the following command to edit the configuration:

```
vim /etc/fluent-bit/fluent-bit.conf
```

b. Replace the contents of the file with the following information:

- For a local Syslog:

```

[SERVICE]
    flush          1
    daemon         Off
    log_level      info
    parsers_file   parsers.conf
    plugins_file   plugins.conf
    http_server    Off
    http_listen    0.0.0.0
    http_port      2020
    storage.metrics on

[INPUT]
    Name tail
    Path /var/log/containers/*.log
    Parser cri
    Mem_Buf_Limit 5MB
    Skip_Long_Lines On

[OUTPUT]
    name stdout
    match *

```

- For a remote Syslog:

Replace the variables in the [OUTPUT] section with values for your deployment:

```

[SERVICE]
    Daemon Off
    Flush 1
    Log_Level info
    Parsers_File parsers.conf
    HTTP_Server On
    HTTP_Listen 0.0.0.0
    HTTP_Port 2021
    Health_Check On

[INPUT]
    Name tail
    Path /var/log/containers/*.log
    multiline.parser docker, cri
    Tag kube.*
    Mem_Buf_Limit 5MB
    Skip_Long_Lines On

[INPUT]
    Name systemd
    Tag host.*
    Systemd_Filter _SYSTEMD_UNIT=kubelet.service
    Read_From_Tail On

[FILTER]
    Name kubernetes
    Match kube.*
    Merge_Log On
    Keep_Log Off
    K8S-Logging.Parser On
    K8S-Logging.Exclude On

[FILTER]
    Name lua
    Match *
    Script /etc/fluent-bit/append-tag.lua
    call append_tag

[OUTPUT]
    Name syslog
    Match *
    Host <host>
    port 514
    mode udp
    syslog_format rfc5424
    syslog_maxsize 2048
    syslog_hostname_key hostname
    syslog_hostname_preset <hostname of local machine/log source>
    syslog_appname_key tag
    syslog_message_key log

```

Note: The default values shown in the [OUTPUT] section can be modified as explained in the [fluent-bit documentation](#).

c. Save and close the file.

7. Run the following commands to start the fluent-bit service:
 - a. Enable fluent-bit to start after a system restart:

```
systemctl enable fluent-bit
```

- b. Start fluent-bit:

```
systemctl start fluent-bit
```

API Connect 10.0.5.1: Update the remote logging configuration

To update the fluent-bit configuration, complete the following steps:

Procedure

1. Run the following command to edit the configuration file:

```
vim /etc/fluent-bit/fluent-bit.conf
```

2. Complete your configuration changes, then save and close the file.
3. Run the following command to restart the fluent-bit service:

```
systemctl restart fluent-bit
```

Gathering logs for a VMware environment

The `generate_postmortem.sh` script gathers all logs for troubleshooting and diagnostics.

About this task

When you contact IBM Support, logs and VMware environment data are normally required to help diagnosis. The `generate_postmortem.sh` script gathers all the required logs and data. The data that is gathered by the script includes OS-level information from each VM, so if you have a three replica deployment, it is recommended to run the `generate_postmortem.sh` script on each VM.

The logs that are gathered by the `generate_postmortem.sh` script include the logs of the currently running API Connect processes. The logs have a maximum size, and older log messages are not retained when the maximum size is reached. To ensure that the logs gathered cover the time when the problem occurred, it is recommended to reproduce the problem and then immediately gather the logs. Offboarding the logs to an external server is the recommended best practice for long-term storage of log data. See [Configuring remote logging for a VMware deployment](#).

Procedure

1. Connect to the target appliance with SSH then switch to the `root` user with the following commands:

```
ssh apicadm@{ova appliance hostname}
sudo -i
```

2. Download the `generate_postmortem.sh` script. Enter the following command:

```
curl -s -o generate_postmortem.sh https://raw.githubusercontent.com/ibm-apiconnect/v10-postmortem/master/generate_postmortem.sh
```

Note: If your API Connect VMs do not have external internet access, download the `generate_postmortem.sh` script to another computer that has internet access, and copy it to your API Connect VMs.

3. Add execute permissions to the `generate_postmortem.sh` script. Enter the following command:

```
chmod +x generate_postmortem.sh
```

4. Run the postmortem tool by using the following command:

```
./generate_postmortem.sh --ova --pull-appliance-logs
```

5. Optional: Specify either management, portal, or analytics for more detailed logs from that subsystem. Specify `--diagnostic-all` for detailed logs of every subsystem. Enter the following arguments:

```
Portal: ./generate_postmortem.sh --ova --diagnostic-portal --pull-appliance-logs
Management: ./generate_postmortem.sh --ova --diagnostic-manager --pull-appliance-logs
Analytics: ./generate_postmortem.sh --ova --diagnostic-analytics --pull-appliance-logs
```

Note: The diagnostic level logs generated with this argument produce a much larger output file.

6. If you see errors when you run the `generate_postmortem.sh` script, run again with `--debug`:

```
./generate_postmortem.sh --ova --debug &>debug.log
```

For more documentation on the script, see <https://github.com/ibm-apiconnect/v10-postmortem>.

Changing logging levels

You can enable logging for entry and exit trace and for large payloads for apim-v2 pods.

About this task

For apim-v2 pods, the default logging settings do not include entry and exit trace and logging of large payloads.

Logging of large payloads is typically needed if you are logging `apim:webhookPayload` or other processes that log large variables.

You can also set the debug level without needing to restart the pods.

You can enable the settings through either the APIM deployment YAML file, the toolkit, or the REST APIs. Note, you must use the toolkit or REST API to set the debug level without restarting the pods.

Procedure

- Updating the settings in the APIM deployment YAML file

Note: This action causes a pod restart.

- The `DEBUG` variable works the same as prior releases.

```
- env:
  - name: DEBUG
    value:
audit,bhendi:error,bhendi:probe,bhendi:flags,apim:server,apim:error,apim:routes:*,apim:routesc:*,apim:oidc,apim:oidc:*
```

To enable entry and exit trace, add `trace:*` to the start of the debug string:

```
- env:
  - name: DEBUG
    value: trace:*,audit,bhendi:error,bhendi:probe,bhendi:flags,bhendi:webhookAudit,bhendi:cassandra-
transactions,apim:server,apim:error,apim:routes:*,apim:routesc:*,apim:oidc,apim:oidc:*,apim:taskmanager:info
```

- To enable large object logging:

```
- env:
  - name: VELOX_LOG_FULL_PAYLOAD
    value: "true"
```

- Updating the settings by using the toolkit

Note: This action does not cause a pod restart. Also, this action is for a single APIM pod environment only.

```
apic log-spec:get
apic log-spec:update LOG_SPEC_FILE
```

Where the spec file contains:

```
{"specification": "apim:routes:log_spec,audit,bhendi:error,apim:server,apim:error", "large_objects": true}
```

- Updating the settings by using the REST API

Note: This action does not cause a pod restart. Also, this action is for a single APIM pod environment only.

```
GET /cloud/api/log-spec
PUT /cloud/api/log-spec
```

For example, to set debug level and large objects logging:

```
curl -k https://172.16.140.212:3003/api/cloud/log-spec -X PUT -H "Authorization: Bearer $BEARER"
-H "Accept: application/json" -d '{"specification":
"apim:routes:log_spec,audit,bhendi:error,apim:server,apim:error","large_objects": true}'
-H "Content-Type: application/json"
```

The command responds with:

```
{
  "specification": "apim:routes:log_spec,audit,bhendi:error,apim:server,apim:error",
  "large_objects": true,
  "message": "Successfully changed the log specification on all apim-v2 pods. Success on IP(s): 172.16.140.212,
172.16.140.213"
}
```


Note: The response might include the following warning, which can be ignored:

```
WARNING: Could not change the log specification on all apim-v2 pods. Success on IP(s): 4.5.6.7 Failed on IP(s): 1.2.3.4
```

To view the REST APIs for logging, go to [API Connect REST APIs](#), and select IBM API Connect Platform - Cloud Management API 2.0.0 reference_>_Resource: Log Spec.

Troubleshooting

Troubleshoot your deployment on VMware.

Note: In the  Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.

See:

- [Recreating a calico pod that is in the CrashLoopBackOff state](#)
If you complete an upgrade and find that one or more of the calico pods are in the `CrashLoopBackOff` state, you can delete the pod to recreate it.
- [Running a filesystem check on a VMware root partition](#)
You can run a filesystem check on the appliance root filesystem at boot time.
- [Troubleshooting the management database](#)
You can troubleshoot the API Connect management database
- [Increasing the PostgreSQL archive timeout on VMware](#)
Increase the `archive_timeout` to 3600 (seconds) so that accumulated wal files do not cause a disk-space issue during a network outage.
- [Dynamically re-registering and reconfiguring a Gateway service in a VMware deployment](#)
In API Connect, Gateway services do not persist their configuration settings by default. Instead, the master configuration is stored on the Management server and the *Dynamic Reregistration and Reconfiguration* (DRR) mechanism resynchronizes configuration data when needed. The DRR process is used when proper High Availability/Disaster Recovery (HA/DR) is not configured, or if a manual resynchronization is required.
- [Troubleshooting Analytics on VMware](#)
If your API Connect Analytics subsystem encounters problems, review the following topics for help resolving the issue.

Recreating a calico pod that is in the `CrashLoopBackOff` state

If you complete an upgrade and find that one or more of the calico pods are in the `CrashLoopBackOff` state, you can delete the pod to recreate it.

Complete the following steps:

1. Get the names of the calico pods:

```
kubectl -n kube-system get pods | grep calico
```

2. Recreate each pod by deleting it with the following command:

```
kubectl -n kube-system delete pod <calico_pod_name>
```

Running a filesystem check on a VMware root partition

You can run a filesystem check on the appliance root filesystem at boot time.

About this task

If the appliance root filesystem is mounted read-only because it is corrupt, use the following steps to run `fsck` on root partition:

Procedure

1. Access to the VMware console for the Appliance, and after causing it to reboot, make sure to bring up a console to the appliance (as needed to give focus to the window by clicking the mouse into it), and press the `Esc` key repeatedly.
You will see a boot screen similar to the following:

```
GNU GRUB v2.02
.
- Ubuntu
- Advanced options for Ubuntu
.
.
Press Enter to boot the selected OS, 'e' to edit the commands
before booting or 'c' for a command-line
```

2. Press `e` to edit the commands before booting, and add the lines `fsck.mode=force` and `fsck.repair=yes`.
For example, the boot screen contents resemble:

```
GNU GRUB v2.02
.
set params 'Ubuntu'
fsck.mode=force
fsck.repair=yes
.
.
Press Ctrl-x or F10 to boot
.
```

3. Press `F10` to boot.
4. The following command should show that `fsck` was run recently:

```
root@ip-230:~# sudo tune2fs -l /dev/sda1 | grep "Last checked"
Last checked: Wed Jul 1 19:32:25 2020
```

Troubleshooting the management database

You can troubleshoot the API Connect management database

Database Replica Pods stuck in Unknown or Pending state

In certain scenarios, a postgres replica pod may not recover to a healthy state when a restore completes, a node outage occurs, or after a fresh install or upgrade. In these cases, a postgres pod remains in a **Unknown** or a **Pending** state after a number of minutes. The pod fail to get into a **Running** state.

This situation occurs when the replicas do not initialize properly. You can use the `patronictl reinit` command to reinitialize the replica. Note that this command syncs the replica's volume data from the current Primary pod.

Use the following steps to get the pod back into a working state:

1. SSH into the appliance as `root`.
2. Exec onto the failing pod:

```
kubectl exec -it <postgres_replica_pod_name> -n <namespace> -- bash
```

3. List the cluster members:

```
patronictl list
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) -----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| fxpk-management-01191b80-postgres-586f899fdf-6s25b | 172.16.172.244 | | start failed | | unknown |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51 | Leader | running | 3 | |
| fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m | 172.16.53.68 | | running | 3 | 0 |
+-----+-----+-----+-----+-----+-----+
```

In the example shown above `fxpk-management-01191b80-postgres-586f899fdf-6s25b` is not in running state.

Note the `clusterName` and `replicaName` which are not up:

- `clusterName` - `fxpk-management-01191b80-postgres`
- `replicaName` - `fxpk-management-01191b80-postgres-586f899fdf-6s25b`

4. Run:

```
patronictl reinit <clusterName> <replicaName-which-is-not-running>
```

Example:

```
patronictl reinit fxpk-management-01191b80-postgres fxpk-management-01191b80-postgres-586f899fdf-6s25b
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) -----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| fxpk-management-01191b80-postgres-586f899fdf-6s25b | 172.16.172.244 | | start failed | | unknown |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51 | Leader | running | 3 | |
| fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m | 172.16.53.68 | | running | 3 | 0 |
+-----+-----+-----+-----+-----+-----+
Are you sure you want to reinitialize members fxpk-management-01191b80-postgres-586f899fdf-6s25b? [y/N]: y
Success: reinitialize for member fxpk-management-01191b80-postgres-586f899fdf-6s25b
```

5. Run `patronictl list` again.

You may also observe that the replica is on a different Timeline (TL) and possibly have a Lag in MB. It may take a few minutes for the pod to switch onto the same TL as the others and the Lag should slowly go to 0.

For example:

```
bash-4.2$ patronictl list
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) -----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| fxpk-management-01191b80-postgres-586f899fdf-6s25b | 172.16.172.244 | | running | 1 | 23360 |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51 | Leader | running | 3 | |
| fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m | 172.16.53.68 | | running | 3 | 0 |
+-----+-----+-----+-----+-----+-----+
```

6. The pod that previously was in an **Unknown** or **Pending** state or **(0/1) Running** state is now in **(1/1) Running** state.

Increasing the PostgreSQL `archive_timeout` on VMware

Increase the `archive_timeout` to 3600 (seconds) so that accumulated wal files do not cause a disk-space issue during a network outage.

About this task

PostgreSQL sends its wal archives to the backup location (S3 or local PVC) whenever the configured `archive_timeout` period elapses. By default, the `archive_timeout` is set to 60 seconds. During a network outage, the files cannot be transferred and instead accumulate on the current server, and could cause a disk-space issue. Increasing the `archive_timeout` value helps to mitigate this issue by creating archive files less frequently.

For information on tracking the Postgres disk space, see [Monitoring Postgres disk usage on VMware](#).

Complete the following steps to increase the PostgreSQL `archive_timeout` value.

Procedure

- SSH into the server by completing the following steps:
 - Run the following command to connect as the API Connect administrator, replacing `ip_address` with the appropriate IP address:


```
ssh ip_address -l apicadm
```
 - When prompted, select Yes to continue connecting.
 - When you are connected, run the following command to receive the necessary permissions for working directly on the appliance:


```
sudo -i
```
- Get the name of the `pgha` config map by running the following command:


```
kubectl -n <namespace> get cm -l pgha-config='true'
```

The response looks like the following example:

NAME	DATA	AGE
apis-prod-d18f3f0-0d18f3f0-postgres-pgha-config	5	22d
- Edit the config map by running the following command:


```
kubectl -n <namespace> edit configmap <config-map-name>
```

In the command, replace `<config-map-name>` with the `NAME` returned in step 1.
- Change the `archive_timeout` value to `3600`.
The value must be an integer, and represents the number of seconds that must elapse before the archive file is generated.
- Save the file and exit the editor.
The configuration is updated automatically; archives will be generated and pushed every `3600` seconds.

Dynamically re-registering and reconfiguring a Gateway service in a VMware deployment

In API Connect, Gateway services do not persist their configuration settings by default. Instead, the master configuration is stored on the Management server and the *Dynamic Reregistration and Reconfiguration* (DRR) mechanism resynchronizes configuration data when needed. The DRR process is used when proper High Availability/Disaster Recovery (HA/DR) is not configured, or if a manual resynchronization is required.

If a Gateway service is not configured properly for resiliency and is restarted, the gateways in the Gateway service will lose the configuration from the Management server. Configuration data from the Management server is maintained on the gateway service according to the gateway peering configuration on the gateways.

Preventing the loss of configuration data

For high availability in production environments, use a minimum of three gateways in the Gateway service. In non-production environments, if a single gateway is used, then availability can be improved by ensuring the persistence setting on the gateway peering configuration is set to `local://` or `RAID`; these settings apply to virtual and physical appliances only.

Forcing re-population of the configuration data

To repopulate the configuration data, complete the steps in the following section to trigger an outage-less DRR.

Triggering an outage-less DRR

To force a Dynamic Reregistration and Reconfiguration across a peer group from within the gateway service without requiring a restart, complete the following steps using the CLI. This process flushes the primary peering object configured for the `apic-gw-service`. This task can only be performed with CLI, and requires the default admin ID for running the `diag` command.

- Use `ssh` to connect to gateway service.
- Switch to the API Connect application:


```
switch <APIC_APP_DOMAIN>
```
- Show the current gateway-peering-status:


```
show gateway-peering-status
```
- If the server to which the ssh connection was made is not primary for the `apic-gw-service` gateway-peering object, force this server to become primary:


```
config; gateway-peering-switch-primary <gateway-peering-name-for-apic-gw-service>;
```
- Flush the `apic-gw-service` gateway-peering object:


```
diag; gateway-peering-flush <gateway-peering-name-for-apic-gw-service>; exit
```
- Disable and then re-enable the `apic-gw-service` object for every member of the gateway service:


```
config; apic-gw-service; admin-state disabled; exit
apic-gw-service; admin-state enabled; exit; exit
```
- Confirm that the `apic-gw-service` was flushed and is now waiting for gateway service registration:
Look in the debug log target for the `apic-gw-service`, located in `logtemp://` and check for a message similar to the following example:


```
20200618T232339.332Z [0x88e000d7][apic-gw-service][notice]
apic-gw-service(default): tid(2653): Waiting for gateway service registration.
```

The DRR will be triggered by the next arrival of a webhook event. The Management server sends a heartbeat to the gateway at five-minute intervals, prompting the gateway to check whether it has lost its configuration and if so, trigger a DRR. In addition, if the Management server has previously marked a Catalog or cloud as being unavailable for a particular gateway service, a successful heartbeat triggers synchronization for that Catalog or cloud on that gateway service.

8. (Optional) Force a BAU webhook event to be sent from API Manager to trigger the DRR:

If you do not want to wait for the Management server to send a heartbeat, you can trigger the DRR manually by completing the following steps:

- a. Open the API Manager.
- b. Open the Catalog.
- c. Click Settings > Edit.
- d. Update the Summary field with some text so that it is modified.
- e. Click Save.

When the event is received, the DRR is initiated.

Troubleshooting Analytics on VMware

If your API Connect Analytics subsystem encounters problems, review the following topics for help resolving the issue.

- [Analytics offload issues on VMware](#)
If your Analytics offload fails, you can perform checks to help determine the cause. Common causes for an offload failure are connection or security-related issues.
- [Empty Analytics display on VMware](#)
If your API Connect analytics data doesn't display in the Analytics dashboard, complete the following steps to determine the cause of the problem.
- [Analytics running out of disk space on VMware](#)
If your Analytics data is taking up too much disk space, complete the following steps to resolve the problem.

Analytics offload issues on VMware

If your Analytics offload fails, you can perform checks to help determine the cause. Common causes for an offload failure are connection or security-related issues.

When you save the configuration files, their syntax is first validated to verify that it can be applied. However, this does not confirm that other parts of your analytics collection ecosystem are trouble-free, for example, that your endpoints are running, or that the TLS certificates are validated. Errors in these areas may also result in no data output.

To verify issues you will need to examine the logs of your analytics ingestion pods. (Note that it may take a few minutes for the pods to pick up changes in the configuration files after edits are applied.)

For information on obtaining logs, refer to [Logging](#).

Following are some common checks you can make to troubleshoot the offload:

- Verify that the specified URL or the host and port values for the target system are accurate.
- Verify that the server or cluster of servers in the target system are running and reachable. If applicable, also verify that the connection problems are not caused by firewall settings.
- If you are using Transport Layer Security (TLS) to establish a private and secure communication channel to the target system, investigate whether the issue is caused by SSL certificate errors; for example, untrusted server certificates, certificates that have expired or are not yet valid, or missing intermediate (or chain) certificates. For more information on using TLS, see [Creating a TLS profile](#).
- If offloading to an Elasticsearch or Kafka target, verify that your configuration settings are valid; for example, the user credentials for authenticating to Elasticsearch or the Kafka topic name.

Empty Analytics display on VMware

If your API Connect analytics data doesn't display in the Analytics dashboard, complete the following steps to determine the cause of the problem.

Before you begin

This operation requires the use of the Analytics CLI. [Using the Analytics CLI](#).

Procedure

1. SSH into the server:

- a. Run the following command to connect as the API Connect administrator, replacing *ip_address* with the appropriate IP address:

```
ssh ip_address -l apicadm
```

- b. When prompted, select Yes to continue connecting.

- c. When you are connected, run the following command to receive the necessary permissions for working directly on the appliance:

```
sudo -i
```

2. Run the following command to check the logs of the mtls-gw pod:

```
kubectl logs analyt-mtls-gw-7665f5d445-4cx4n -n <APIC_namespace>
```

The response looks like the following example:

```
10.254.6.1 - - [22/Sep/2021:18:05:54 +0000] "POST /x2020/v1/apievent/_bulk HTTP/1.1" 200 2 "-" "-" 2841 0.072 0.070
```

- a. Check the response for any `_bulk` request coming from the gateway, as shown in the example.
If there are no `_bulk` requests, then the problem is with the gateway and the `wsm-agent`, which is not sending bulk requests to the Analytics server.
- b. If you see the `_bulk` request., then check the return code in the response.
If `_bulk` requests are processed successfully you will see a return code 200 (as in the example), which indicates that the bulk request has been ingested.
- c. If it is any other return code, check the ingestion pod logs for any errors, which will indicate any issues during ingestion:

```
oc logs analyt-ingestion-78d7cff7f7-6pktd
```

3. If everything with the data flow looks fine, and if you are unable to see analytics data, then check the health of analytics using the CLI:

```
apic --mode analytics clustermgmt:getHealth --server <management server platform api endpoint> --analytics-service <analytics service name> --format json
```

where:

- `<management server platform api endpoint>` is the Management server admin endpoint.
- `<analytics service name>` is the name of Analytics service as configured in the Cloud Manager.

When you check the cluster's health, the response for a single node analytics deployment looks like the following example:

```
{
  "cluster_name": "apic-analytics-cluster",
  "status": "yellow",
  "timed_out": false,
  "number_of_nodes": 1,
  "number_of_data_nodes": 1,
  "discovered_master": true,
  "discovered_cluster_manager": true,
  "active_primary_shards": 8,
  "active_shards": 8,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 12,
  "delayed_unassigned_shards": 0,
  "number_of_pending_tasks": 0,
  "number_of_in_flight_fetch": 0,
  "task_max_waiting_in_queue_millis": 0,
  "active_shards_percent_as_number": 40
}
```

Note: Single node analytics clusters never show **green** in the health output. Green indicates that the data is replicated to at least one other node, and since there are no other nodes in a single node cluster it remains as **yellow**.

- a. Check the status of the cluster (red, yellow, green) in the response.
In the example, the status is set to yellow.
 - Green indicates all indices are fine.
 - Yellow indicates there are some unassigned shards, which might still be replicating.
 - Red indicates there is a problem with one or more of the indices.
- b. Run the following command to get a list of indices:

```
apic --mode analytics clustermgmt:catIndices --server [SERVER] --analytics-service [ANALYTICS SERVICE]
```

where:

- `SERVER` is the Management server admin endpoint.
- `ANALYTICS SERVICE` is the name of Analytics service as configured in the Cloud Manager.

- c. From the list of indices returned, identify the those with status of 'red'.
- d. Delete each red index by running the following command:

```
apic --mode analytics clustermgmt:deleteIndex --index <index_name> --server [SERVER] --analytics-service [ANALYTICS SERVICE] --format json
```

where:

- `SERVER` is the Management server admin endpoint.
- `ANALYTICS SERVICE` is the name of Analytics service as configured in the Cloud Manager.

Analytics running out of disk space on VMware

If your Analytics data is taking up too much disk space, complete the following steps to resolve the problem.

Before you begin

This operation requires the use of the Analytics CLI. [Using the Analytics CLI](#).

About this task

Analytics indices taking up too much space

Procedure

1. Run the following command to display the indices:

```
apic --mode analytics clustermgmt:catIndices --server [SERVER] --analytics-service [ANALYTICS SERVICE]
```

where:

- **SERVER** is the Management server admin endpoint.
- **ANALYTICS SERVICE** is the name of Analytics service as configured in the Cloud Manager.

2. From the output of above command, identify the old and large indices that can be deleted, and delete each index using the following command:

```
apic --mode analytics clustermgmt:deleteIndex --index <index_name> --server [SERVER] --analytics-service [ANALYTICS SERVICE] --format json
```

where:

- **SERVER** is the Management server admin endpoint.
- **ANALYTICS SERVICE** is the name of Analytics service as configured in the Cloud Manager.

3. Reduce the retention period: [Configuring data retention and index rollover time periods](#)

4. Apply a filter to reduce the index size is by excluding large fields, such as **response_body**, from being saved in the index.

For example:

```
filter {
  if "apicapievent" in [tags] {
    mutate {
      remove_field => ["response_body"]
    }
  }
}
```

For information on creating filters for Analytics data, see [Modifying your analytics data](#).

Migrating from API Connect v5 Public Cloud

Prepare to move to API Connect v10 by migrating your existing data from API Connect v5 Public Cloud.

The migration process provides tools to extract v5 data, arrange it for v10 formats, and load it onto a v10 deployment.

What can I migrate to v10?

The resources and assets associated with your v5 Public Cloud instance are included in the migration; for example, TLS profiles, APIs, Catalogs, Products, Consumers, subscriptions, user registries, and developer organizations. In addition, your billing data (Stripe integrations, paid Plans, and consumer payment methods) is migrated from v5 to v10.

Analytics data (including searches, visualizations, and dashboards) cannot be migrated due to changes in the Analytics feature. If you want to preserve your event data, you should export it from v5 Public Cloud using either [CSV files](#) or [REST API calls](#) as explained in the v5 Public Cloud documentation.

Important: The migration utility allows migration from v5 to a v10 v5-compatible gateway or v10 API Gateway. References to conversion of v5-compatible artifacts to v10 API Gateway refer to the v5-compatible artifacts that were initially from a v5 environment and were unpacked by the migration utility.

How do I migrate to v10?

You should be familiar with v5 Public Cloud architecture and requirements, and with the deployment configuration that you want to migrate to v10. The migration process requires you to have administrator-level access to both v5 Public Cloud and to your v10 on-premises deployment. During the migration process you can keep your existing v5 system active while you set up your v10 API Connect deployment.

The migration process involves the following high-level tasks, which are described in the topics that follow:

1. Deploy API Connect v10.
2. In your v5 Public Cloud instance, enable migration and download your data.
3. Install the AMU (API Connect Migration Utility) tool on the computer where you downloaded your data.
4. Unpack the downloaded data, which creates a directory structure containing YAML files where you can modify data for use in v10.
5. Use the AMU tool to upload the modified data to your v10 API Connect deployment.
6. Recreate billing integrations and other customizations as needed.

- [Migration considerations](#)

You can optimize the migration process by reviewing the migration considerations and taking any actions that apply to your deployment.

- [Preparing to migrate from v5 Public Cloud](#)

Plan the migration, deploy API Connect v10, download v5 Public Cloud data, and set up the migration utility.

- [Migrating data and Portal Delegated User Registries](#)

Complete the following steps to migrate v5 Public Cloud data and Portal Delegated User Registries.

- [Migration troubleshooting](#)

If you run migration and observe error messages in your log output, you can take troubleshooting steps, and run the migration again.

Migration considerations

You can optimize the migration process by reviewing the migration considerations and taking any actions that apply to your deployment.

Review the following information:

- [Limitations and workarounds](#)
- [Version 5 data that is not migrated](#)
- [Parity between v5 and v10](#)

- [Customizations to recreate](#)

Limitations and workarounds

- When migrating catalog spaces in a provider organization that uses a local user registry (LUR), ensure that the LUR is configured with the visibility setting of either Public or Custom. If set to Custom, ensure that the provider organization is selected.
- Migration of the v5 username token policy for authenticating the username token is not supported.
- In v10, you cannot publish a product containing an API with a `regex` Swagger property that includes the (or ? in the characters for the pattern. You can expect to see an error message similar to the following example:

```
(? The preceding token is not quantifiable
```

- The migration utility validates Products and APIs against the OpenAPI specification (validation is not performed on draft Products and draft APIs). The `archive:unpack` command issues warnings for any Product or API that fails validation. To resolve warnings, fix the Product or API and re-run the unpack command. Example warning:

```
WARN[0019] Error validating product cloud/provider-orgs/onlinebanking/catalogs/test/products/clientipfilter-policy-test-api-1.0.0.yml:
Validation errors:
Error validating API cloud/provider-orgs/onlinebanking/catalogs/test/products/api123-1.0.0.api.yml
- Must validate one and only one schema (oneOf) (context: (root).securityDefinitions, line: 36, col: 3)
- securityDefinitions.authorizationUrl Does not match format 'uri' (context: (root).securityDefinitions.authorizationUrl, line: 0, col: 0)
```

- OAuth Provider APIs:
 - To ensure that APIs that use OAuth security definitions are mapped to the corresponding OAuth Provider APIs in v10, make sure that either the Authorization URL or the Token URL defined in OAuth Security Definitions of APIs that use OAuth correctly point to the base path of the OAuth Provider API. Note that the Migration tool uses that logic to map Consumer APIs to Provider APIs, and adds `x-ibm-oauth-provider` in the OAuth Security Definition of the consumer API.
 - To verify whether OAuth Provider APIs are migrated correctly, ensure that all the APIs that use OAuth Security Definition have the new field `x-ibm-oauth-provider` added. Check the value of that field and make sure it points to a correct OAuth Provider resource object that was created and configured in the catalog to which the API is published.
 - In v5, OAuth Providers are a specialized type of API that are published within products. In v10, OAuth Providers are first class objects that are directly published in the provider organization and are not APIs that are published within products. Because of this, it is possible that you may have some v5 products that only contained OAuth Provider APIs. Since these products do not contain any Open API definitions in v10, they will not be published as all products in v10 must contain at least one Open API definition.

- Portal Delegated User Registry migration limitations
 - OIDC: Automated migration is not supported for some v5 PDUR OIDC registries and you will see instructions for providing the `client_id` and `client_secret` while mapping the registries.
 - LDAP: Automated configuration is not supported for LDAP registries and you will see instructions for configuring them manually while mapping the registries.
- The SCIM (System for Cross-domain Identity Management) user registry type was supported in Version 4 but was deprecated for Version 5 and is not supported in v10. If data with a SCIM identity provider type is detected during migration of v5 data, the migration utility logs the following error message:

```
WARN[0614] manual fix required for cloud/provider-orgs/apimteam/user-registries/scimcv/user-registry.yml:
The SCIM user registry type is no longer supported
```

Any v5 data with a SCIM identity provider type must be manually migrated. For v10, you can use OIDC with a back-end user registry that supports SCIM. You must develop custom code to complete the data flow between the OIDC provider and your backend user registry. The API Connect built-in OIDC registry interacts with your OIDC provider.

- Roles limitations

In API Connect v5, it was possible to create a role (and associated permissions) at the organization level and the role was inherited by the catalogs within the organization. With the v5 inheritance model it was possible to modify the inherited role and add or remove permissions.

In API Connect v10 the inheritance model has changed and it is no longer possible to remove permissions from an inherited role. Because of this design change, it becomes difficult to perform a true 1-to-1 migration keeping the original parent-child relationships. For example, in v5 it was possible for an inherited role to contain fewer permissions than its parent role, but this is not supported in v10. To work around this restriction, the parent-child relationship is purposely decoupled during migration, and a new artificial parent role is created for each of the child roles.
- Modifications made to default roles in v5 are not migrated to v10.
- If the v5 settings `server_name_indication` and `limit_renegotiation` are not specified, the value defaults to `true`. However if they are specified with value of `false` that value is changed to `true`. Thus the settings are always `true`.
- Subscriptions on staged products are not automatically migrated because in v10, staged products cannot have subscriptions on them. To save those subscriptions, change the state of the product to either deprecated or published before running the `archive:push` command to upload data to v10. You can do this either in v5 before downloading your data, or in by mapping or editing the downloaded product metadata and to change the state of the product. If you have already pushed the staged product, then you can publish the product in your v10 environment and rerun `archive:push` to migrate the subscriptions.
- The migration utility appends `-apigw` to the end of the `x-ibm-name` of APIs to maintain uniqueness from the original v5 artifacts. Therefore, custom backend services that rely on the `x-ibm-name` of the APIs to be matched must be updated to look for `-apigw` in the new API name. Alternatively, the custom backend service can be modified to strip `-apigw` from the API name.
- When migrating default catalog user registries, and migration has completed, you might see more than one catalog User Registry with the same title. Whenever a new catalog is created in v10, API Connect creates a default catalog user registry. When migration takes place, the catalog user registry from v5 is migrated onto the v10 system. Both these registries receive the same Title; however they receive different Names and function as separate user registries (registries are managed by Name, not Title).

Example: If the catalog is called `test`:

- Name of user registry migrated from v5: `test-catalog`
- Name of default user registry created in v10: `test-catalog-0`
- Title of both the user registries: `Test Catalog User Registry`

All the user data is successfully migrated to the user registry coming from v5; manually update the catalog settings to use the migrated v5 user registry as the default. You can optionally clear the default user registry provided by v10 from the list of configured catalog user registries.

If you manually create the catalog and users already exist in the default catalog user registry provided by v10, then both the migrated v5 catalog user registry and the v10 default user registry can coexist, and either of them can be selected as default.

Version 5 data that is not migrated

- APIs with LTPA policies are not migrated because this policy is not supported in v10.
- Configuration in v5 for gateway type **Microgateway** is not migrated because **Microgateway** is not supported in v10.
- Transient or conflicting configuration data is not migrated:
 - Tasks that are still active are not included in the downloaded v5 data; for example, lifecycle approvals or subscription approvals that have been requested but not granted.
 - Drafts on v5 that are local are not migrated. If you have draft APIs or draft products that you want migrated to v10, use `apic drafts:push` to upload in v5 before you download your v5 data for migration. For more information, see [Working with drafts](#) in the v5 Public Cloud documentation.
 - Pending invitees in user registries are not migrated.
 - Pending invitations for ownership of provider organizations are not migrated; however consumer organizations with pending ownership are migrated. If the v5 provider organization owner received an invitation for ownership but never accepted it, the owner identity is not migrated, which causes the organization to not be migrated. To avoid this issue, check for any unaccepted invitations prior to migration, and ensure they are accepted.
 - Conflicting user names within the same identity provider are not migrated.
 - Conflicting OAuth provider names or base paths are not migrated. If the migration utility encounters conflicting names or base paths for OAuth providers, the provider is not migrated.
- OAuth Shared secret keys are not migrated unless you explicitly [map them in the mapping.yml file for the gateway](#).
- Organizations without an active owner (user identity) are not migrated
One effect of this is that if the original owner of the organization is an identity that was not migrated, the provider organization or consumer organization cannot be migrated. Prevent this issue by verifying that the owner of the organization is a current user.
- Analytics data is not migrated
Analytics data (including searches, visualizations, and dashboards) cannot be migrated due to changes in the Analytics feature. If you want to preserve your event data, you should export it from v5 Public Cloud using either [CSV files](#) or [REST API calls](#) as explained in the v5 Public Cloud documentation.

Parity between v5 and v10

- The migration utility sluggifies (a slug is a human-readable, unique identifier) the names and versions of some elements when migrating from v5 to v10 to ensure that each element in v10 passes Open API schema validation and is uniquely named. The conversion rules differ depending upon which version you are migrating to.
 - **Version 10.0.5.1 and earlier:** For the names of API properties, Catalogs, Provider Organizations, Consumer Organizations, Spaces, and some other elements, any character other than `a-z`, `0-9`, or `-`, is converted to `-` and a unique suffix (based on the checksum of the name or version) is appended. Uppercase letters are changed to lowercase, with `-` added between words. Leading `-` characters are removed, and multiple consecutive `-` characters are reduced to a single `-`. For versions of similar elements, the `.` (period) character is also allowed.
 - **Version 10.0.5.2 and later:** For the names and versions of API properties, Products, Consumer Organizations, Space and some other elements, any character other than `a-z`, `A-Z`, `0-9`, `-`, `_`, or period (`.`), is converted to `-` and a unique suffix (based on the checksum of the name or version) is appended. Catalog and Provider Organization names follow the same rules, except that they do not support the period (`.`) character. Names or versions with uppercase characters are not converted to lowercase, but a unique suffix is appended to them.

If you are using the latest version 10.0.5.2 or later version of the migration utility, you can use the `--use-old-sluggify` parameter with the `apicm archive:unpack` command to unpack the V5 configuration data onto the local filesystem using the version 10.0.5.1 conversion rules. You can use this option in the following scenarios.

- You are starting a new migration to API Connect v10.0.5.1 with version 10.0.5.2-R4 or later of the migration utility.
- You started the migration using version 10.0.5.1 of the migration utility and have upgraded to version 10.0.5.2-R4 or later.

For more information, see the migration instructions that apply to your deployment.

- [Migration steps with PDUR](#)
- [Migration steps without PDUR](#)

Table 1. Example migrated names on v10

v5 name	Migrated name on v10.0.5.1 or earlier	Migrated name on v10.0.5.2 or later
abc-def	abc-def (no change)	abc-def (no change)
Abc	abc-a2c6	Abc-a2c6
AbcDef	abc-def-dfc	AbcDef-dfc
Abc Def	abc-def-1e82 With <code>sluggify-no-checksum</code> : abc-def	Abc-Def-1e82 With <code>sluggify-no-checksum</code> : Abc-Def
abc_def	abc-def-c2bb	abc_def (no change)
abc--def	abc-def-813d	abc--def (no change)
-abc-def	abc-def-abb	-abc-def (no change)
ABC-DEF	abc-def-d1e	ABC-DEF-d1e

Table 2. Example migrated versions on v10

v5 version	Migrated version on v10.0.5.1 or earlier	Migrated version on v10.0.5.2 or later
1.0.0	1.0.0 (no change)	1.0.0 (no change)
1.0_2	1.0-2-3933	1.0_2 (no change)
1.0 2	1.0-2-913	1.0-2-913
1.0--2	1.0-2-5410	1.0--2 (no change)
-1.0-2	1.0-2-26cf	-1.0-2 (no change)

You can use `apicm sluggify <string>` to view how a name or version string will be slugged. The following examples are for version v10.0.5.2 or later:

```
$ ./apicm sluggify Abc
Abc-a2c6
$ ./apicm sluggify -v 1.0_2
1.0_2
$ ./apicm sluggify --version '1.0 2'
1.0-2-913
```

- In v10, Spaces are not supported in the Sandbox Catalog.
Spaces in Sandbox Catalogs are not migrated from v5 to v10. The names of Spaces found in v5 Sandbox Catalogs are logged in the migration output log. To preserve the Spaces, use a mapping file to rename the Sandbox Catalogs.

- In v10, publishing an API fails if enforced is set to `true`, and phase is set to either `Specified` or `Identified` instead of `Realized`. During migration, the phase will be set to `Realized` to allow the API to be published.
- Enforcement of user subscription to an API is handled differently in v5 and v10. In v5, subscription to a product is not allowed when in the following scenario: An API is created with enforced set to `false`, the API is associated with a product, the product is published, and then a user attempts to subscribe to a Product or Plan from the Developer Portal. In v10, the subscription is allowed for this scenario. This change can affect migration because a v5 Product that could not be subscribed to in v5 can be subscribed to in v10. Note that the change in behavior for v10 accommodates the scenario where Products or Plans have multiple APIs, some of which are enforced and some of which are not enforced. The new behavior allows such Products or Plans to be subscribed to.
- In v10, the API Manager handles all authentication methods. This change eliminates the need for configuration of a catalog with a Portal Delegated User Registry (PDUR), so during migration you will map PDURs to v10 standard user registries (see [Migration steps with PDUR.](#))
- If `OpenAPI.info.contact.email` contains more than one email address, only the first email address is migrated. If the email is not in valid RFC5322 email format then it is removed during migration.
- Provider-side applications:
In API Connect, registered applications are a fundamental concept representing an application identity known to the system. Applications have their own credential sets in the form of client ID and client secret pairs, and access to published API endpoints is governed by subscriptions associated with registered applications. Most commonly, registered applications are created by app developers belonging to a consumer organization in the context of a developer portal.

However, API Connect v5 also provides the ability for provider organizations to register and manage applications, independent of any consumer organization. Provider-owned apps behave in the same way as consumer-owned apps and carry credentials and subscriptions in exactly the same way. Customers use provider-owned applications for API testing or for cases where the app and API are owned and managed directly by the same team.

The API Connect v10 platform API does not support the creation of provider-side apps, so migration of provider-side apps is only possible if they are converted to consumer-side apps. Each provider-side app is migrated as follows by the API Connect migration utility:

- An instance of the app including its credentials (sets of client ID and secret) is created in each catalog belonging to the provider organization.
- The app is owned by a consumer organization which is the existing Sandbox Test Organization (for the provider organization's automatically created sandbox catalog) or a consumer organization named `Catalog-name Test Organization` (for other catalogs).
- The consumer organization's owner is a user named `test-user`. This user name already exists in the context of a provider organization's automatically created sandbox catalog.
- The consumer organization's owning user belongs to a user registry named `catalog-name-test-org`.
- The subscriptions belonging to the original provider-side app are retained.

These definitions are added to the relevant locations subordinate to the `cloud/` directory that is produced by the `apicm archive:unpack` command. When an `apicm archive:push` is performed, the definitions are propagated to the target system as part of the overall configuration.

Customizations to recreate

Version 5 provides methods to customize some configurations. After you complete the automated migration, manually reapply the following customizations.

- Customized email templates
Customized email templates are not migrated from Version 5. You can customize email templates in Version 10 with greater flexibility and precision. See [Customizing email templates.](#)
- Developer Portal customizations are not migrated to a new Drupal version
For the Developer Portal, none of the customizations are migrated because the v10 Drupal-9 baseline differs greatly from the v5 Drupal-7 baseline. This means that custom portal themes, custom modules, custom fields, custom blocks, custom pages, and more cannot be migrated. For more information, see [Migrating your Developer Portal from Version 5 to Version 10.](#)

Preparing to migrate from v5 Public Cloud

Plan the migration, deploy API Connect v10, download v5 Public Cloud data, and set up the migration utility.

Procedure

Complete the following steps:

1. Plan your migration:
 - a. Review all of the migration topics to ensure you understand the requirements, the general process, and important considerations.
 - b. Plan your new v10 deployment.
When moving from v5 to v10, use the opportunity to review your current resources, such as user registries, provider organizations, and so on. You might want to look at the existing data in v5 Public Cloud and remove or consolidate data before running the extract. This will help ensure only objects that are still needed in v10 will be migrated, and that your data will require less clean-up in the new deployment.
 - c. Plan to do run several test iterations of migration.
You can use the migration utility's `--dry-run` option to perform iterations of the migration push without actually uploading data to v10. The test runs offer an opportunity to test for errors and warnings that need to be resolved before you perform the actual data push.
 - d. Prepare a pilot migration of your APIs.
API Connect v10 on-premises supports both a v5-compatible DataPower Gateway and the newer DataPower API Gateway, which has different requirements.

To ensure a smooth migration, convert 10-to-20 of your most commonly used v5 APIs to meet the requirements of the API Gateway. Use one or more test runs to ensure that APIs do not generate errors during migration. Then, migrate the converted APIs and test them in v10 to verify that they can be used in the new environment, and to determine the appropriate migration utility settings before you convert your remaining APIs.
2. [Install and configure API Connect v10.](#)
Configure v10 with usage of the incoming v5 data in mind. Before you push your v5 data to v10, your v10 system must be up and running, with all required gateway services, provider organizations, user registries, and so on, ready to receive the data. In particular, ensure that your v10 deployment meets the following requirements for migration:

- Understand the requirements for the Gateway in v10:
API Connect v10 supports two types of gateway services: DataPower Gateway (v5-compatible) and DataPower API Gateway (the DataPower API Gateway type is not available in API Connect v5). When you register your gateway service on your v10 system, you must select a gateway service type (see [API Connect gateway types](#)).

If you select the API Gateway, then you must complete the following tasks:

- Ensure that **xm1-manager** is enabled in the default domain of all gateways.
- Your migration process must include a step to convert the v5 API Definitions from the DataPower Gateway format to the DataPower API Gateway format. For information on converting the v5 APIs, see [Converting APIs to DataPower API Gateway](#).
- Review the OAuth Security Definitions of APIs that use OAuth.
To ensure that APIs that use OAuth security definitions are mapped to the corresponding OAuth Provider APIs in v10, make sure that either the Authorization URL or the Token URL defined in OAuth Security Definitions of APIs that use OAuth correctly point to the base path of the OAuth Provider API. Note that the migration utility uses logic to map Consumer APIs to Provider APIs, and adds **x-ibm-oauth-provider** in the OAuth Security Definition of the consumer API.
- Prepare a provider organization in v10 that will own the v5 Public Cloud assets:
In v5 public cloud, there is a single provider organization; all of your data migrates to a single provider organization in v10. If the target provider organization in v10 uses the same name as in v5 public cloud, data is migrated automatically. If the names don't match, you can map the names while preparing your data for migration (see [Mapping v5 resource names to v10](#)).
- Prepare user registries in v10:
Configure user registries to match the user registries in v5 Public Cloud. The migration process migrates users and sends email to users about their accounts.

Note: Portal Delegated User Registries are not supported in v10 because they are replaced with equivalent registries that are managed by the Management subsystem. If you used the PDUR feature in v5 Public Cloud, create matching registries in v10 so that you can migrate users to the new registries. For information on mapping a PDUR registry to a v10 registry, see [Mapping v5 PDUR registries to v10](#).
- Create the Portal services that you want to use with the Catalogs that you plan to migrate:
You will configure the catalogs to the Portals after the migration data is pushed to the v10 system.


3. Set up the AMU migration utility on a secure computer where you will process the data from V5 Public Cloud before importing it into V10.
The V5 Public Cloud migration archive file must be downloaded to the same computer, so that you can use the AMU to push the V5 Public Cloud data to your v10 deployment.

Note: When running the migration utility commands on Windows, open a command window with Administrator privileges, and omit the "/" from the commands.

- a. Install Node 18.17.1 (or greater).
To use the migration utility for converting API definitions for the API Gateway, you must have Node version 18.17.1 (or greater) installed.
- b. Download and extract the newest version of the [AMU for API Connect 10.0.5](#) from IBM Fix Central.
Note: If you do not already have entitlement to use IBM Fix Central, contact IBM Support to request access.
When you unpack the data from v5 Public Cloud using the AMU tool unpack command, it is stored in a directory called **c1oud** within the same folder structure where the AMU is stored. When you extract the AMU, keep the following considerations in mind:
 - Keep the path to the AMU short, especially on Windows.
Windows has path length limits that can interfere with large extracts. To avoid this issue, keep the path for your extracted data as short as possible (short folder names with as flat a folder structure as you can manage).
 - Do not include a folder named **c1oud** in the path where you extract the AMU.
When you unpack the data from v5 Public Cloud, it is stored in a folder called **c1oud** within the same folder structure where the AMU is stored. If a folder named **c1oud** already exists, the conflict will cause an error during the unpack process.
- c. Start the AMU:
 - i. In the folder where you extracted the AMU, navigate to the appropriate subfolder for your operating system (Mac, Linux, or Windows).
 - ii. Open a command window where you can run AMU commands.
Note: On Windows: open a command window with Administrator privileges.
 - iii. Verify that the utility is available by running the **apicm** command to view the list of available commands; for example:

```
./apicm
```


Note: On Windows: omit the "/" from the command.

 Run the "apicm" command to see the full list of commands

Tip: Use the **apicm <command-name> --help** command to review the command options and their purpose. To see the list of commands, run: **apicm help**.

4. Get owner access to the v5 Public Cloud provider organization.
Only the provider organization's owner can download the data for migration. If you will migrate data but are not the owner of the provider organization, contact the current owner and have that person complete the following steps to make you the new owner.

Attention: Only the current owner of the provider organization can transfer ownership to another user.

- a. Log in to v5 Public Cloud as the owner of the provider organization.
- b. Click Admin,Members.
- c. In the list of members, locate the row containing the name of the person who will become the new owner of the provider organization.
- d. On the row containing the selected user, click  Actions menu,Transfer Organization Owner.
- e. At the confirmation message, click OK.

Notes:

- The previous owner of the provider organization loses access to the Admin,Members page.
- The previous owner is removed from the provider organization unless they also owned a catalog (limitation in v5). As the new owner, you can add that person as a member of the provider organization.
- Transferring ownership of the provider organization does not transfer ownership of any catalogs or spaces. The previous provider organization owner still retains ownership of all catalogs and spaces that they previously owned.

5. Download your v5 Public Cloud data.

Complete this step on the secure computer where you have the AMU migration utility installed.

- a. Log in to V5 Public Cloud as the owner of the provider organization.
Only the owner of the provider organization can migrate data.

- b. Open the Dashboard, locate the "API Connect Public Cloud instances will expire soon" message and click Download.
- c. At the "Select the type of API Connect environment to which the assets will be migrated" message, select API Connect v10 on-premise installation as your migration target, and click Next.
- d. At the "Please enter API Connect Service Instance URL from AWS Subscriptions and Services Instances View Details Page" message, type or paste the URL for accessing your instance of API Connect Enterprise as a Service, and click Continue.
You can obtain the URL from the link to your instance that is provided on the IBM Automation management console's [Subscriptions](#) page.
- e. At the "Please enter an encryption password then press the download button which will start preparing the data" message, create an encryption password:
The encryption password that you create here is required for extracting the downloaded data. If you lose the password you cannot retrieve it; however you can download the data again with a new password.
 - i. In the Encryption Password field, type a new password.
 - ii. In the Re-enter Encryption Password field, type the new password again to confirm it.
- f. At the "Assets Download" message, click the Download button to start the data extraction from v5 Public Cloud.
It might take a few minutes for the assets to be prepared and the button to be enabled.
- g. At the "Downloading is in progress" message, click OK.
The extracted data is downloaded as an archive file called `<ProviderOrgName>-extract.tgz`. Proceed to the next step to extract the data.

6. Decompress the archive file containing the v5 Public Cloud data.

Run the following command to decompress the `<ProviderOrgName>-extract.tgz` that you downloaded from v5 Public Cloud:

```
tar xvf "<ProviderOrgName>-extract.tgz"
```

The decompress process creates the following files:

- `extract-backup.ecr` contains your v5 Public Cloud data. If you did not use the PDUR (Portal Delegated User Registry) feature in v5, this file also contains your registry information.
- (PDUR only) `pdurexport.ecr` If you used PDUR feature in v5 Public Cloud, then registries were managed by the Portal subsystem and this file contains the registry information from the v5 Developer Portals.

7. Unzip the decompressed files using the encryption password that you created when you downloaded the archive.

- a. Unzip the `extract-backup.ecr` file by running the following command and replacing with your encryption password:

```
unzip -P "<encryption_password>" extract-backup.ecr
```

This step creates the `extract-backup.tgz` file.

- b. If available, unzip the `pdurexport.ecr` file by running the following command and replacing with your encryption password:

The `pdurexport.ecr` file is only available if you used the PDUR feature in v5.

```
unzip -P "<encryption_password>" pdurexport.ecr
```

This step creates the `pdurexport.tgz` file.

8. Use the AMU to unpack the `extract-backup.tgz` and `pdurexport.tgz` files.

The AMU must be run in a command window with administrative privileges on Windows. You might prefer to run the migration utility from a Linux VM on your Windows computer.

Run the following command to unpack the files (if you do not have a `pdurexport.tgz` file, omit it from the command):

```
./apicm archive:unpack ./extract-backup.tgz ./pdurexport.tgz
```

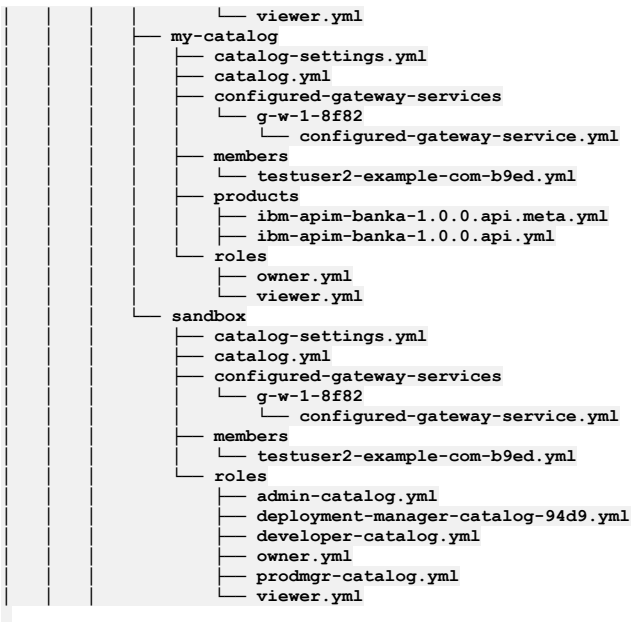
The output from this command looks like the following example:

```
INFO[2022-06-20T13:03:16-07:00] Saving copy of: saveErrorLog to: logs/archive-unpack20220620T13-03-16error.log
INFO[2022-06-20T13:03:16-07:00] Saving copy of: saveInfoLog to: logs/archive-unpack20220620T13-03-16info.log
INFO[2022-06-20T13:03:16-07:00] Disabled PDUR errors as no PDUR archive was passed in
INFO[2022-06-20T13:03:16-07:00] unpacking v5 archive ./extract-backup.tgz
INFO[2022-06-20T13:03:23-07:00] Writing unpacked content to /Users/single_porg/osx/cloud
.....
INFO[2022-06-20T13:03:39-07:00] Completed unpacking of provider_org_name organization contents.
INFO[2022-06-20T13:03:39-07:00] Completed unpacking archive ./extract-backup.tgz.
Admin Org Stats:
- Availability Zones: 1
.....
Provider Org Stats:
- Provider Organizations: 1
.....
```

The unpack process produces YAML files containing your v5 data formatted for use in v10 and stores them in a folder structure stemming from the `cloud` folder.

Example output from the unpacked migration data:

```
cloud
├── provider-orgs
│   └── <providerOrg>
│       ├── catalogs
│       │   └── develop
│       │       ├── catalog-settings.yml
│       │       ├── catalog.yml
│       │       ├── configured-gateway-services
│       │       │   └── g-w-1-8f82
│       │       │       └── configured-gateway-service.yml
│       │       ├── members
│       │       │   └── testuser2-example-com-b9ed.yml
│       │       ├── products
│       │       │   ├── account-1.0.0.api.meta.yml
│       │       │   └── account-1.0.0.api.yml
│       │       └── roles
│       │           └── owner.yml
```



Troubleshooting:

- Path: The unpack command might return an error message stating that the `cloud` must be empty, as in the following example:

```
ERRO[2022-06-20T13:02:00-07:00] AMUAU000E: The migration tool failed to initialize.
Reason: The cloud directory /Users/single_porg/osx/cloud must be empty to run the unpack command
```

The error indicates that the location where you extracted the files already contained a folder named `cloud`, which caused a conflict. Correct the error by extracting the files to a folder that does not already contain a subfolder named `cloud`.

- Invalid characters: The unpack command ensures that catalog names and API properties contain only valid characters; invalid characters are replaced with `"_"`.
- Validation issues: The unpack command issues warning messages for APIs and Products that fail validation against the OpenAPI specification. Correct the APIs and Products before attempting to migrate them into v10.
- The unpack command might return error messages related to your PDUR archive missing some information, as in the following example:

```
ERRO[3414] PDUR backup data for IDP PortalDelegatedIdentityProvider-123b81fbe4b06de411580e2e
(123b81fbe4b06de411580e2f) not provided: not found
ERRO[3414] PDUR backup data for IDP PortalDelegatedIdentityProvider-123b820de4b06de411580e32
(123b820de4b06de411580e33) not provided: not found
```

If your v5 Public Cloud instance used the PDUR feature, this error indicates that the AMU could not find all of the specified Portal user registries when it unpacked the `pdurexport.tgz` archive. To resolve this problem, download your data from V5 Public Cloud again, and then re-run the unpack command.

If you are not migrating PDUR data and your command did not include the `pdurexport.tgz` file, this error indicates that you do in fact have a PDUR registry, and that you should have included the `pdurexport.tgz` file in the unpack command. To resolve this problem, download your data from v5 Public Cloud again and re-run the unpack command with the `pdurexport.tgz` file included.

- Any other Warnings or Errors that occur during `archive:unpack` process should be investigated for remediation (see [Migration troubleshooting](#)).

Migrating data and Portal Delegated User Registries

Complete the following steps to migrate v5 Public Cloud data and Portal Delegated User Registries.

About this task

Use the migration utility to migrate your API Connect v5 Public Cloud data to a v10 deployment. Your v10 deployment must be up and running, with gateway services, provider organizations, user registries, and so on, ready to receive the migrated data.

Procedure

1. Optional: Set up a temporary SMTP server to hold automatic emails without sending them during migration. During migration, if subscription approval is needed to enable an application to subscribe to a plan, notifications are sent to all users within the organization who have subscription approval permission. In some cases, this results in a large number of emails. To avoid sending these emails, set up a temporary email server prior to running the migration, and use it to collect all notifications rather than sending them to users' email accounts.

If subscription approvals are not needed during your migration, you can skip this step.

2. Optional: Set up a temporary LDAP server. A temporary LDAP server can be needed if you are temporarily working behind a firewall.

- a. Install a server:

```
git clone https://github.com/veo-labs/ldap-server-mock
```

```
npm install
```

b. Creating the configuration files:

- `ldap-server-mock-conf.json`

```
{
  "port": 3004,
  "userLoginAttribute": "cn",
  "searchBase": "dc=test",
  "searchFilter": "(&(objectclass=person)(cn={username}))"
}
```

- `users.json`

```
[
  {
    "dn": "cn=user,dc=test",
    "cn": "user-login"
  }
]
```

c. Run the LDAP server:

```
node server.js --conf=./ldap-server-mock-conf.json --database=./users.json
```

d. Modifying a current LDAP provider.

The following file is a dummy user registry file that maps to a read-only test server. The fields in bold font (`admin_dn` and `endpoint`) have been changed to fit the dummy server configuration. Change these on all user registries that need to be fixed.

```
api_version: 2.0.0
configuration:
  admin_dn: cn=user,dc=test
  admin_password: xxxx
  authenticated_bind: "true"
  authentication_method: search_dn
  bind_prefix: ""
  bind_suffix: ""
  protocol_version: "3"
  search_dn_base: ""
  search_dn_filter_prefix: (examplelegacyuid=
  search_dn_filter_suffix: )
  search_dn_scope: sub
endpoint:
  endpoint: ldap://172.20.20.11:3004
  tls_client_profile_url: idurl://...\\admin-org\\tls-client-profiles\\tls-profile-for-ldap-1.0.0.yml
integration_url: file://...\\integrations\\user_registry\\ldap.yml
metadata:
  v5_id: xxxxxx40cf2fc09a9ba0d6c
  v5_scope: ["api"]
  v5_type: ldap
name: xxxxxxxxxxxxxxxx
title: xxxxxxxxxxxxxxxxxxxxxxxx
type: user_registry
```

3. Map v5 resource names to v10 resources names as explained in [Mapping v5 resource names to v10](#).

Most deployments adjust the names or quantities of gateway services, provider organizations, and user registries when setting up a new environment for v10. Be sure to specify any changed v10 destinations for the v5 data in a migration mapping file so that the migration utility can automatically move the data.

4. If you used the PDUR feature in v5 Public Cloud, map your Portal Delegated User Registries to the equivalent registries that you created in v10, as explained in [Mapping v5 PDUR registries to v10](#).

API Connect v10 supports new registries and you no longer need the PDUR feature. Instead, you can configure functionally equivalent registries in v10 that are maintained by the Management subsystem and migrate users to the new registries.

5. Convert the v5 API definitions to formats required by v10 DataPower API Gateway as explained in [Migrating v5 APIs to v10 API Gateway format](#).

Note: If you are migrating your APIs to a DataPower v5-compatible Gateway, skip this step because there is no need to convert your APIs.

6. Remove unneeded configuration objects.

Inspect the set of configuration objects and their attributes, and remove the ones that are not needed. In many cases, some of the configuration objects present in the v5 system are not needed in the new v10 environment. For example, there is no value in preserving inactive users, obsolete apps, obsolete consumer organizations, and so on.

When deleting objects, make sure that you do not delete objects that have other dependent objects, such as products with subscriptions on them or users that are active. If you delete a consumer organization, make sure you also delete all apps and subscriptions associated with that consumer organization. Be aware of any groups or visibility settings that include these consumer organizations.

7. Upload the v5 data to v10.

To upload the v5 data, log in to your v10 deployment with the migration utility and push the data with the migration utility. The admin organization and its data, and then the provider organization and all of its assets, are imported into v10. You can push to v10 as many times as needed to correct errors and provide missing information.

a. Get the log-in information from your v10 deployment.

Log in to your v10 deployment and note down the `<v10_host_name>` from the URL:

For example, in the following URL, the v10 host name is `apicdev10.acme.com`:

```
https://apicdev10.acme.com/manager/
```

b. On the computer where your extracted v5 Public Cloud data is stored, use the migration utility to log in to your v10 deployment as an API Connect administrator:

Note: On Windows: open a command window with Administrator privileges, and omit the `"/` from the command.

```
./apicm login -s <v10_host_name> -r admin/default-idp-1 -u admin
```

For example:

```
./apicm login -s apicdev10.acme.com -r admin/default-idp-1 -u admin
```

c. Use the migration utility to push your v5 Public Cloud admin organization data to the v10 deployment:

In the same directory where you ran the `apicm: archive:unpack` command, run the following command where `<v10_host_name>` is the host name:

```
./apicm archive:push <v10_host_name> cloud/admin-org
```

For example:

```
./apicm archive:push apicdev10.acme.com cloud/admin-org
```

The output from this command looks like the following example:

```
INFO[2022-06-22T12:54:25-07:00] Saving copy of: saveErrorLog to: logs/archive-push20220622T12-54-25error.log
INFO[2022-06-22T12:54:25-07:00] Saving copy of: saveInfoLog to: logs/archive-push20220622T12-54-25info.log
.....
INFO[2022-06-22T14:46:35-07:00] archive:push for organization admin completed
```

d. On the same computer, use the migration utility to log in to your v10 deployment's provider organization:
Log in as an Owner of the provider organization.

```
./apicm login -s <v10_host_name> -r provider/default-idp-2 -u provider_org_owner
```

For example:

```
./apicm login -s apicdev10.acme.com -r provider/default-idp-2 -u example_user
```

e. Use the migration utility to push your v5 Public Cloud provider organization data to your v10 deployment:

In the same directory where you ran the `apicm: archive:unpack` command, run the following command where `<v10_host_name>` is the host name and `<v10_provider_organization_name>` is the name of the provider organization:

```
./apicm archive:push <v10_host_name> cloud/provider-orgs/<v10_provider_organization_name>/
```

For example:

```
./apicm archive:push apicdev10.acme.com cloud/provider-orgs/my-provider-org/
```

Table 1 describes optional parameters that you can include when you push your v5 Public Cloud provider organization data to your v10 deployment.

Table 1. Optional parameters for the `apicm archive:push` command

Optional parameter	Description
<code>--catalogs=<cat_1,cat_2,cat_n></code>	Push data for only the specified Catalogs (if you don't want to push the entire provider organization). When a catalog is mapped, specify the v10 Catalog name.
<code>--dry-run</code>	Create a test directory similar to <code>cloud</code> (but titled <code>.workdir-<apim_admin_server></code>) where you can view how the data will be loaded into your new system. This option also displays the objects that will fail during migration, providing an opportunity to correct errors before actually importing data into v10 (see Migration troubleshooting).
<code>--no-drafts</code>	Do not push any draft (unpublished) APIs or Products. This option can optionally be used with the <code>--catalogs</code> parameter. This option cannot be used with the <code>--drafts-only</code> option.
<code>--drafts-only</code>	Only push draft (unpublished) APIs or Products and do not push any published APIs or Products. This option cannot be used with <code>--catalogs=a,b,c</code> or the <code>--no-drafts</code> option.
<code>--overwrite</code>	Overwrite all existing assets in the v10 deployment. Omit this parameter if you want to retain existing data in v10; for example, if you run the migration in iterations.
<code>--apigw-only</code>	Optional flag. Use when migrating to a DataPower API Gateway.

The output from this command looks like the following example:

```
INFO[2022-06-22T12:54:25-07:00] Saving copy of: saveErrorLog to: logs/archive-push20220622T12-54-25error.log
INFO[2022-06-22T12:54:25-07:00] Saving copy of: saveInfoLog to: logs/archive-push20220622T12-54-25info.log
.....
INFO[2022-06-22T14:46:35-07:00] archive:push for organization <v10_provider_organization_name> completed
```

Troubleshooting:

- The `apicm` login token expires after eight hours. If the token expires while `apicm archive:push` is running, `apicm` generates errors due to lack of the necessary access permissions. There are two workarounds:
 - Log in again and re-run `apicm archive:push`. The token is refreshed with each new login.
 - Extend the token expiration time by logging in again with an optional argument: `-expireHours=<number_of_hours>` (see [Token expiration](#)).
- If you observe error messages in your log output, consult the [Migration troubleshooting](#) topic for information on resolving the problems, and then run the migration again. You can push to v10 as many times as needed to correct errors and provide missing information.

8. When migration successfully finishes, complete the following steps:

a. Apply any customized configurations from your v5 deployment (see [Customizations](#)).

b. Optional: Switch the v10 provider organization ownership to the identity that owned it in v5.

This step might be necessary if you created and migrated the provider organization using a different identity (such as administrative account) than the user identity of the v5 provider org. Note that the UI does not provide this function, but you can use the API `/orgs/{org}/transfer-owner` to switch ownership.

c. If Stripe billing integrations were included in your migration, configure the Stripe integrations in your v10 Developer Portal sites; see [Configuring Stripe in the Developer Portal](#) for detailed instructions. For more information about the billing feature in v10, see [Monetizing your Products](#).

- [Mapping v5 resource names to v10](#)

Create files to map the name of each v5 Public Cloud resource to the corresponding v10 resource.

- [Mapping v5 PDUR registries to v10](#)

Map your Portal Delegated User Registries to equivalent v10 registry types.

- [Migrating OAuth providers and OAuth secured APIs](#)
You can migrate v5 OAuth providers and OAuth secured APIs to your v10 API Connect deployment.
- [Migrating v5 APIs to v10 API Gateway format](#)
You can convert API Connect v5 APIs to the format required by the DataPower API Gateway in v10.
- [Logging in to a management server with an OIDC registry](#)
Logging in with an OIDC registry involves several steps, in which you obtain a temporary token and pass it to the toolkit to ensure it can communicate securely with the management server.

Mapping v5 resource names to v10

Create files to map the name of each v5 Public Cloud resource to the corresponding v10 resource.

About this task

The names of some API Connect resources that you used in v5 Public Cloud might be different in v10. Mapping files allows you to specify the v10 resource that corresponds to a particular v5 resource that uses a different name, ensuring that data associated with the resource can be migrated. You only need to create mapping files for resources that use a different name in the old and new versions of API Connect.

Note: If you used Portal Delegated User Registries in v5 Public Cloud, you will map those registries to v10 registries in a later task.

What does a mapping file do?

Mapping files are simple YAML files that you create in the directory structure created by the `apicm archive:unpack` command. Any directory that contains a YAML file describing a configuration object might need a mapping file to specify the name of the corresponding v10 resource.

A mapping file contains the name of the v5 resource file that it is related, to plus one or more **key: value** pairs that specify the v10 values to apply in place of the original v5 configuration values.

What kinds of settings can I map?

You can override any values in the YAML configuration files; the most common mappings for v5 migrations are for the Gateway service name and the provider organization name.

What does a mapping file contain?

A mapping entry consists of two or more lines:

- The first line in the mapping file specifies the name of the v5 configuration file that you want to override, but without the `.yaml` extension. For example, when overriding a value in `provider-org.yaml`, the first line of the mapping is `provider-org`.
- Additional lines specify **key: value** pairs that override the V5 **key: value** pairs that appear in the V5 file that you are overriding. The most common reason for mapping is to override the resource name; for example, to map a V5 provider organization to a new name in V10.
To override the resource name, include the **name:** setting as the key, and the new V10 resource name as the value. The following example replaces the V5 provider organization name in the V5 `provider-org` file (located in the current folder) with the name `myv10porg` in V10:

```
provider-org:
  name: myv10porg
```

Where do I create mapping files?

Create each v10 mapping file in the same folder where the corresponding v5 configuration file is stored, within the cloud folder structure that was created when you unpacked the v5 Public Cloud files. Create one mapping file in each folder, and name every mapping file with the following filename: `mapping.yaml`.

In the following steps, you will create 2 mapping files: one for the resources associated with v5 Public Cloud provider organization and one for the resources associated with the v5 Public Cloud Gateway service.

Procedure

Map the provider organization resources:

1. Locate the `provider-org.yaml` in the unpacked cloud folder.
 - a. In the unpacked cloud folder, locate the `provider-orgs` subfolder.
 - b. In the `provider-orgs` folder, locate the folder containing the `provider-org.yaml` file.
2. In the same folder as the `provider-org.yaml` file, use any text editor to create a new file called `mapping.yaml`.
3. Map the name of the v5 Public Cloud provider organization to the v10 Enterprise as a Service provider organization.
Example `mapping.yaml` file for the provider organization (note the indentation):

```
provider-org:
  name: <v10_provider_organization_name>
```

where `<v10_provider_organization_name>` is the name of the provider organization in v10 that you want to migrate the v5 assets to.

4. Save and close the provider organization `mapping.yaml` file.



Map the gateway service resources:

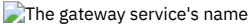
5. Locate the `gateway-service.yaml` in the unpacked cloud folder.
 - a. In the unpacked cloud folder, locate the `configured-gateway-services` subfolder.
 - b. In the `configured-gateway-services` folder, locate the folder containing the `gateway-service.yaml` file.
6. In the same folder as the `gateway-service.yaml` file, use any text editor to create a new file called `mapping.yaml`.
7. Map the v5 DataPower Gateway service to the v10 DataPower API Gateway service.
 - a. Add the following content to the `mapping.yaml` file to map the v5 Gateway service to the v10 API Gateway service:
Example `mapping.yaml` file for the Gateway service (note the indentation):

```
gateway-service:
  name: <v10_gateway_service_name>
```

```
gateway-type: datapower-api-gateway
```

where `<v10_gateway_service_name>` is the name (not the Title or ID) of the v10 DataPower API Gateway service. To determine the gateway service name, complete the following steps:

- i. Log in to API Connect v10 API Manager and click , `<catalog_name>`, Catalog settings, Gateway services.
- ii. On the Gateway services page, click  Actions menu icon next to the gateway service and select View status.
- iii. On the Gateway processing status page, choose any item in the "Event" column, locate the "(name: *value*" that follows the event description, and copy the value (shown highlighted in the following image).

The gateway service's name

Leave the mapping.yml file open for the next step.

8. Map a migrated OAuth shared secret key.

Information about migrated OAuth shared secret keys is contained in your `gateway-service.yml` file, so . When you map a v5 gateway to v10, you must ensure that the OAuth shared secret key is associated with the v10 gateway name.

Note: Only migrate one OAuth shared secret key; mapping more than one key to a service will result in an error.

- a. Determine whether the v5 Gateway service has an OAuth shared secret associated with it.

In the v5 `gateway-service.yml` file, look for a key called `oauth_shared_secret:`. If your file does not contain this key, then no OAuth shared secrets were migrated and you can skip to step [9](#).

- b. If the file does include the key, then add `migrate_oauth_shared_secret: true` to the `mapping.yml` file.

Example mapping.yml file for the Gateway service and OAuth shared secret (note the indentation):

```
gateway-service:
  name: <v10_gateway_service_name>
  gateway-type: datapower-api-gateway
  migrate_oauth_shared_secret: true
```

Leave the mapping.yml file open for the next step.

9. Map Server Name Indication (SNI) settings.

When enabled, SNI settings (which include the host and TLS Server Profile pairs) are included in the migration of gateway services.

- a. Determine whether the v5 Gateway service has SNI settings to migrate. your `gateway-service.yml` file contains SNI settings to migrate

In the v5 `gateway-service.yml` file, look for a section called `sni:` containing one or more `host:` keys. If your file does not contain this section, then no SNI settings were migrated and you can skip to step [10](#).

- b. If the file does include the `sni:` section, then add `migrate_sni: true` to the `mapping.yml` file.

Example mapping.yml file for the Gateway service, OAuth shared secret, and SNI settings (note the indentation):

```
gateway-service:
  name: <v10_gateway_service_name>
  gateway-type: datapower-api-gateway
  migrate_oauth_shared_secret: true
  migrate_sni: true
```

10. Save and close the gateway service `mapping.yml` file.

Mapping v5 PDUR registries to v10

Map your Portal Delegated User Registries to equivalent v10 registry types.

Before you begin

If you did not use the PDUR (Portal Distributed User Registry) feature in v5 Public Cloud, then skip this task.

If you used the PDUR feature in v5 Public Cloud, create matching registries in v10 so that you can migrate users from the PDUR registries to the new v10 registries.

About this task

If you used the Portal Delegated User Registry (PDUR) feature for a Catalog in API Connect v5 Public Cloud, then the user management for that Catalog was delegated from the Management subsystem to the Developer Portal and the site administrator within the Portal site performed the configuration tasks. The Management subsystem recorded that the delegation had taken place, and stored a record of the users that existed in the catalog; however, the Management subsystem did not maintain information about the configuration of the specific registries or how users mapped to them.

API Connect v10 supports new registries and you no longer need the PDUR feature. Instead, you configure functionally equivalent registries in v10 that are maintained by the Management subsystem. To migrate the PDUR user registries and users, you will use the `pdurexport.tgz` archive (that you previously downloaded and unpacked) to configure the PDUR registries for v10.

Table 1 shows the mapping Portal Delegated User Registries to the v10 equivalents:

Table 1. Registry type mappings

v5 PDUR Registry	v10 Equivalent Registry	Manual migration required?
LDAP	LDAP	Yes
OIDC	OIDC	Yes
HybridAuth	OIDC	Yes

Procedure

1. If you are migrating a PDUR LDAP user registry, complete the following steps.

The user registries that are required in the target v10 deployment are described in the `user-registry.yml` file for each registry. When possible, the configuration for the v10 registries is complete in these files; however, for LDAP the registry cannot be configured automatically due to differences between how the v5 Portal subsystem and the v10 Management subsystem specify the user registry configuration. Instead, the v5 configuration is provided as helpful information in the `metadata.v5_config` property.

- a. Configure the LDAP user registry for v10.

Configure the v10 user registry by modifying the v5 `user-registry.yml` file and adding the required LDAP fields that are described in the `metadata.v5_config` property. Remove the metadata when the changes are complete.

The `user-registry.yml` file is located in the following directory:

```
cloud/provider-orgs/<v5-provider-org>/user-registries/<PDUR_LDAP_User_registry_name>
```

For example:

```
cloud/provider-orgs/test-user1-org1/user-registries/example-5c38556de4b0c5f69db90ee0/user-registry.yml
```

- b. In the same folder, create a mapping file entry for the LDAP user registry.

Create a file called `mapping.yml` and specify the name of the v10 LDAP user registry that will contain the users from your v5 PDUR:

```
user-registry:  
  name: <v10_LDAP_user_registry_name>
```

Where `<v10_LDAP_user_registry_name>` is the name of the target v10 registry.

2. If you are migrating a PDUR HybridAuth or PDUR OIDC user registry, complete the following steps.

API Connect v10 supports OAuth 2.0 but does not support OAuth 1.0, so OIDC registries that use OAuth 1.0 with v5 Public Cloud cannot be migrated to v10.

OIDC registries that use OAuth 2.0 must be migrated manually by completing the following steps.

- a. Add the `client_id` and `client_secret` to the OIDC user registry.

If your OIDC registry used a `client_id` and `client_secret` in v5 Public Cloud, the values were not included in the downloaded registry information so you must add them now. If you did not use a `client_id` and `client_secret` for the registry, skip this step.

Edit the v5 `user-registry.yml` file and insert the values into the `client_id` and `client_secret` keys. The `user-registry.yml` file is located in the following folder:

```
cloud/provider-orgs/<v5-provider-org>/user-registries/<PDUR_OIDC_User_registry_name>
```

For example:

```
cloud/provider-orgs/test-user1-org1/user-registries/git-hub-5c5xyz6ee4b0c5f69db94xxxx-a339/user-registry.yml
```

Example `user-registry.yml` file for GitHub:

```
api_version: 2.0.0  
configuration:  
  client_id: ""  
  client_secret: ""  
  authorization_endpoint: https://github.com/login/oauth/authorize  
  provider_type: github  
  token_endpoint:  
    endpoint: https://github.com/login/oauth/access_token  
integration_url: file://../../../../integrations/user_registry/oidc.yml  
metadata:  
  v5_id: 5c5xyz6ee4b0c5f69db94xxxx  
  name: git-hub-5c5xyz6ee4b0c5f69db94xxxx-a339  
  registry_type: oidc  
  title: GitHub  
  type: user_registry
```

If your registry requires the `client_id` and `client_secret` but you omit this step, you will see the following errors when you push the OIDC registry to V10 Enterprise as a Service:

```
ERRO[0010] unable to create User registry git-hub-5c5xyz6ee4b0c5f69db94xxxx-a339  
ERRO[0010] git-hub-5c5xyz6ee4b0c5f69db94xxxx-a339: The properties client_id, client_secret are required.
```

- b. Create a mapping file for the OIDC user registry.

Create the mapping file in the following folder (if you modified the `user-registry.yml` file, use the same folder):

```
cloud/provider-orgs/<v5-provider-org>/user-registries/<PDUR_OIDC_User_registry_name>
```

Create a file called `mapping.yml` and specify the name of the migration endpoint for the OIDC user registry used with your V5 PDUR:

```
user-registry:  
  name: <v10_OIDC_user_registry_name>
```

Where `<v10_OIDC_user_registry_name>` is the name of the target v10 registry.

Migrating OAuth providers and OAuth secured APIs

You can migrate v5 OAuth providers and OAuth secured APIs to your v10 API Connect deployment.

In v5 OAuth providers are a specialized type of API that are published within products. In v10 OAuth providers are first-class objects that are directly published in the provider organization. This is applicable for both DataPower Gateway (v5-compatible) and DataPower API Gateway.

Also, v10 adds tight coupling between the API using OAuth and the OAuth provider object. Hence an API using OAuth security must link to the corresponding OAuth provider object. For a review of how API Connect v10 uses OAuth providers, see [OAuth Provider overview](#).

The API Connect Migration Utility (AMU) migrates all OAuth provider APIs from v5 to the first-class OAuth provider objects on both DataPower Gateway (v5-compatible) and DataPower API Gateway.

In addition, the AMU uses logic to determine which OAuth provider is used by a given API and populates it in the newly added field `x-ibm-configuration` of API security definitions. If the AMU is unable to find a match between an API using OAuth and the OAuth provider, it prints out error messages. See [Error messages and warnings from archive:unpack](#).

- To ensure that APIs that use OAuth security definitions are mapped to the corresponding OAuth provider APIs in v10, make sure that either the Authorization URL or the Token URL defined in OAuth Security Definitions of APIs that use OAuth correctly point to the base path of the OAuth provider API. Note that the AMU uses logic to map consumer APIs to provider APIs, and adds `x-ibm-oauth-provider` in the OAuth Security Definition of the consumer API.
- To verify whether OAuth provider APIs are migrated correctly, ensure that all the APIs that use OAuth Security Definition have the new field `x-ibm-oauth-provider` added. Check the value of that field and make sure that it points to a correct OAuth provider resource object that was created and configured in the catalog to which the API is published.
- As mentioned earlier, in v5 OAuth providers are a specialized type of API that are published within products. In v10 OAuth providers are instead first-class objects that are directly published in the provider organization, and are not APIs that are published within products. Because of this, it is possible that you may have some v5 products that only contained OAuth provider APIs. Since these products do not contain any Open API definitions in v10, they will not be published as all products in v10 must contain at least one Open API definition.
- Because v10 OAuth providers are first-class objects rather than APIs as in v5, the application authentication feature is not available to v10 OAuth providers migrated from v5.

Migrating OAuth token revocation information

In v5, token revocation could be enabled either using native token management system (TMS) or using external TMS. For migration to v10:

- Native TMS migration
 - DataPower Gateway (v5-compatible)
If you had configured native token revocation, then you would have configured Quota enforcement peering group objects on v5 to synchronize between gateway servers. When migrating to v5-compatible, a QES peering group can contain QES peers from both v5 Multi-Protocol Gateway Service (MPGW) and v10 DataPower Gateway, so the revocation information is retained.

After setting up a v10 DataPower Gateway instance, follow these steps to configure the v10 DataPower Gateway quota enforcement server (QES) to use the same QES on both v5 and v5-compatible. You must configure the QES and verify that it is enabled and functional before migrating from v5 to v10. For more details about configuring the QES, see the DataPower Gateway documentation topic *Configuring the quota enforcement server*.

1. Log into the graphical interface of the DataPower Gateway instance in the default domain.
2. Cluster the v5 MPGW QES with the v10 DataPower Gateway QES.
 - a. In DataPower Gateway, navigate to Objects > System Settings > Quota Enforcement Server.
 - b. Specify the password alias, data storage location, and the server and monitor port numbers.
If your v10 DataPower Gateway is a Docker instance, configure the QES to store data in memory. For more information about RAID and Docker, see the DataPower Gateway documentation topic *RAID and a DataPower Docker image*.
 - c. Enable Peer group mode.
 - d. Specify the IP address of the DataPower Gateway.
The IP address can be the IP address on any interface of the current appliance and must be accessible by other peers in the peer group.
 - e. Specify the IP addresses of the v5-compatible gateways to add them as peers.
The appliance connects to each peer in the order in which the peers are added in the list.
 - f. Apply and save the configuration.
3. As a test, revoke a token on the v5 MPGW server.
Revocation data will replicate to v10 DataPower Gateway QES.
4. Migrate the v5 APIs to the v10 DataPower Gateway QES.
5. Send a request to the v10 DataPower Gateway QES with previously revoked token, which should fail as expected.
The expected failure indicates that the APIs were migrated and revocation data was replicated.

- DataPower® API Gateway
When migrating to DataPower API Gateway, QES is not used for native token management and revocation. Instead, you can migrate token revocation information from a v5 QES to a gateway-peering cache in v10 DataPower API Gateway. For information, see [Migrating token revocation information to DataPower API Gateway](#).
- [Migrating token revocation information to DataPower API Gateway](#)
You can use the Token Migration Utility (TMU) to migrate token revocation information from a v5 quota enforcement server (QES) to a gateway-peering cache in v10 DataPower API Gateway.
- [Token migration errors](#)
After you run the Token Migration Utility (TMU), check the log in the logs subdirectory for errors. If it contains any of the following error messages, complete the troubleshooting steps before you run the utility again.

Migrating v5 APIs to v10 API Gateway format

You can convert API Connect v5 APIs to the format required by the DataPower API Gateway in v10.

About this task

Converting APIs to the API Gateway format is a two-step process: make a first pass with the migration utility, and then resolve any issues with manual modifications.

Note: If you are migrating your APIs to a DataPower v5-compatible Gateway, skip this step because there is no need to convert your APIs. The migration utility generates an annotated YAML file and logs with information regarding the artifacts that it attempts to convert. Because of differences between the programming models of v5 APIs and v10 API Gateway APIs, the migration utility might not be able to convert all items. In these cases, messages generated by the `apicm archive:port-to-apigw` command provide links to documentation with more details about changes you must make by manually editing the API definitions.

Procedure

1. Automatically convert the APIs with the migration utility by running the following command, where `c1oud` is the directory where the unpacked v5 Public Cloud data resides:

```
./apicm archive:port-to-apigw /c1oud
```

Note: On Windows: open a command window with Administrator privileges, and omit the `./` from the command.

When you run the command, all artifacts within the `c1oud` folder structure are ported to v10 configuration, including APIs, Products, OAuth providers, custom policies, subscriptions, and so on.

Usage notes:

- Specify migration configuration parameters using either a YAML configuration file or with parameters on the command line, but not both. For information about configuration parameters, see [Configuration parameters for migrating APIs](#).
- By default, all changes that the migration utility makes when rewriting APIs are logged as comments within the API YAML file. You can use the `enable-api-logging` parameter to disable logging of these changes, as in the following example:

```
./apicm archive:port-to-apigw /c1oud --enable-api-logging=false
```

- Multiple runs of the command on the same `c1oud` folder will skip artifacts that have already been converted.
- If you decide to rewrite one API at a time, you can replace `c1oud` with the path to the API definition in the command.

With this approach, you cannot use the migration utility to push the API to the v10 deployment. Instead, use the toolkit or the API Manager in v10 to manually import the API definition.

Because of differences between the programming models of v5 APIs and v10 DataPower API Gateway APIs, the migration utility might not be able to convert all items. You might need to port these items to the API Gateway. In these cases, messages generated by the conversion process include links to documentation that provides more details about changes you must make in the next step for these items to be compatible with DataPower API Gateway.

2. Review the conversion results and make any manual changes to the APIs as needed.

The conversion produces an output YAML file, which you can review for information on API conformance, performance, and errors. Review the log files for messages about any manual actions that might be needed. For information on reviewing the results and modifying the migrated APIs, see [Reviewing and correcting migrated v5 APIs](#).

- [Configuration parameters for migrating v5 APIs](#)

If you are converting APIs to DataPower API Gateway, you can accept the default migration configuration or specify configuration parameters.

- [Reviewing and correcting migrated v5 APIs](#)

The migration utility v5 APIs to the format required by the v10 DataPower API Gateway but sometimes encounters issues that require manual intervention.

- [Frequently asked questions about migrating APIs to API Gateway](#)

Answers to some of the most frequently asked questions about migrating APIs to API Gateway.

Configuration parameters for migrating v5 APIs

If you are converting APIs to DataPower API Gateway, you can accept the default migration configuration or specify configuration parameters.

You can specify migration configuration parameters using either a YAML configuration file or from the command line, but not both. The `config.yml` file must be located in the same directory as the migration utility. You can specify to use this file when the migration utility is invoked using the `apicm` command with the `port-to-apigw` option, as in the following example.

```
./apicm archive:port-to-apigw <path_to_API_definition> --use-config-file=true
```

If use of the configuration file is not specified, the migration utility uses parameters specified on the command line. If no parameters are specified, then the default parameters are used.

Table 1. Migration configuration options

Parameter	Valid values	Description
<code>activity-log-policy</code>	<code>activity-log_1.5.0</code> , <code>set-variable_2.0.0</code>	Defines how to process the v5 activity-log policy when the migration utility rewrites the API assembly. The default value is <code>activity-log_1.5.0</code> , which changes the activity-log version to 1.5.0. When set to <code>set-variable_2.0.0</code> , the migration utility sets context variables based upon the value of the activity-log content and error-content configuration values.
<code>chunked-uploads</code>	<code>true</code> / <code>false</code> The default value is <code>false</code> .	If an <code>invoke_1.5.0</code> or <code>proxy_1.5.0</code> policy exists in an assembly in the API, then set <code>chunked-uploads</code> to <code>true</code> to enable chunked encoding for HTTP 1.1 requests to the target server. If an <code>invoke_2.0.0</code> policy exists in an assembly after the migration, then set <code>chunked-uploads</code> to <code>false</code> explicitly for the policy, as the default value for <code>chunked-uploads</code> in API Gateway is <code>true</code> .

Parameter	Valid values	Description
<code>client-id-header-override</code>	String. The string cannot contain spaces. Follow the guidelines for <code>apiKey</code> <code>name</code> :. No default value.	<p>Override the existing <code>X-IBM-Client-Id</code> name value with the new value. In API Gateway, <code>x-key-type</code> makes it possible to define custom naming for <code>client_id</code> and <code>client_secret</code>.</p> <p>Example:</p> <pre>--client-id-header-override="acme-foo"</pre> <p>In this example, the <code>securityDefinitions</code> property is changed from:</p> <pre>foo: type: apiKey name: X-IBM-Client-Id in: header</pre> <p>to the following:</p> <pre>foo: type: apiKey name: acme-foo in: header x-key-type: client_id</pre> <p>Calls to this API must supply <code>acme-foo</code>: <client-id> in the request header instead of the original <code>X-IBM-Client-Id</code>: <client-id>.</p>
<code>client-id-query-override</code>	String. The string cannot contain spaces. Follow the guidelines for <code>apiKey</code> <code>name</code> :. No default value.	<p>Override the existing <code>client_id</code> name value with the new value. In API Gateway, <code>x-key-type</code> makes it possible to define custom naming for <code>client_id</code> and <code>client_secret</code>.</p> <p>Example:</p> <pre>--client-id-query-override="acme-foo"</pre> <p>In this example, the <code>securityDefinitions</code> property is changed from:</p> <pre>foo: type: apiKey name: client_id in: query</pre> <p>to the following:</p> <pre>foo: type: apiKey name: acme-foo in: query x-key-type: client_id</pre> <p>Calls to this API must supply <code>?acme-foo=<client-id></code> in the query parameter instead of the original <code>?client_id=<client-id></code>.</p>
<code>client-secret-header-override</code>	String. The string cannot contain spaces. Follow the guidelines for <code>apiKey</code> <code>name</code> :. No default value.	<p>Override the existing <code>X-IBM-Client-Secret</code> name value with the new value. In API Gateway, <code>x-key-type</code> makes it possible to define custom naming for <code>client_id</code> and <code>client_secret</code>.</p> <p>Example:</p> <pre>--client-secret-header-override="acme-foo"</pre> <p>In this example, the <code>securityDefinitions</code> property is changed from:</p> <pre>foo: type: apiKey name: X-IBM-Client-Secret in: header</pre> <p>to the following:</p> <pre>foo: type: apiKey name: acme-foo in: header x-key-type: client_secret</pre> <p>Calls to this API must supply <code>acme-foo</code>: <client-secret> in the request header instead of the original <code>X-IBM-Client-Secret</code>: <client-secret>.</p>

Parameter	Valid values	Description
<code>client-secret-query-override</code>	String. The string cannot contain spaces. Follow the guidelines for <code>apiKeyName</code> . No default value.	Override the existing <code>client_secret</code> name value with the new value. In API Gateway, <code>x-key-type</code> makes it possible to define custom naming for <code>client_id</code> and <code>client_secret</code> . Example: <code>--client-secret-query-override="acme-foo"</code> In this example, the <code>securityDefinitions</code> property is changed from: <code>foo:</code> <code>type: apiKey</code> <code>name: client_secret</code> <code>in: query</code> to the following: <code>foo:</code> <code>type: apiKey</code> <code>name: acme-foo</code> <code>in: query</code> <code>x-key-type: client_secret</code> Calls to this API must supply <code>?acme-foo=<client-secret></code> in the query parameter instead of the original <code>?client_secret=<client-secret></code> .
<code>copy-id-headers-to-message</code>	<code>true</code> / <code>false</code> The default value is <code>false</code> .	When set to <code>true</code> , the <code>client-id</code> header is copied from <code>request.headers</code> to <code>message.headers</code> to be passed to the backend server. Set this parameter to <code>true</code> if your invoke backend services expect the <code>client-id</code> to be passed through.
<code>custom-policies-scope</code>	<code>catalog</code> / <code>global</code> The default value is <code>catalog</code> .	When set to <code>catalog</code> , the original v5 or v5-compatible catalog that contains the custom policy is staged for pushing to the target catalog and organization in API Gateway. When set to <code>global</code> , the custom policy is staged for pushing to API Gateway and is visible to all organizations and catalogs within that gateway. <code>archive:port-to-apigw</code> sets up the custom policies for pushing to API Gateway, but does not do the pushing. If <code>global</code> scope is used, <code>archive:push</code> of the admin organization will push all custom policies associated with the configured gateways. If <code>catalog</code> scope is used, then the <code>archive:push</code> of the admin organization will initialize the custom policies on the gateway. A subsequent <code>archive:push</code> of the provider organization is needed to advertise the custom policies in the catalogs contained within the provider organization. Note: Potential conflicts between custom policies of the same name and version are resolved according to the following rules: <ul style="list-style-type: none"> • When set to <code>global</code> and two policies under the same gateway service have the same name, version, and md5 hash value, then the duplicates are discarded and a message is generated. • When set to <code>global</code> and two policies under the same gateway service have the same name, version, and a differing md5 hash value, then <code>global</code> scope cannot be applied. <code>catalog</code> scope is used and a warning is generated. • When set to <code>catalog</code>, <code><orgname>_<catname></code> is appended to the zip filename to maintain uniqueness.
<code>deploy-policies</code>	Any combination of the following values: <code>gatewayscript 1.0.0</code> , <code>if 1.5.0</code> , <code>invoke 1.5.0</code> , <code>proxy 1.5.0</code> , <code>redact 1.5.0</code> , <code>switch 1.5.0</code> , <code>validate-usertoken 1.0.0</code> , <code>xslt 1.0.0</code> The default values are: <code>if 1.5.0</code> , <code>invoke 1.5.0</code> , <code>proxy 1.5.0</code> , <code>redact 1.5.0</code> , <code>switch 1.5.0</code> , <code>validate-usertoken 1.0.0</code> , <code>xslt 1.0.0</code>	Defines which backwards-compatible policies should be advertised by DataPower to API Manager. Note: Any legacy policies that are specified in the <code>config.yml</code> file or by using the <code>apicm</code> command must also be specified by using <code>deploy-policies</code> . If these policies are not specified by using <code>deploy-policies</code> , an error is returned.
<code>emulate-v4-plan-rate-limit</code>	<code>true</code> / <code>false</code> The default value is <code>false</code> .	Determines whether all APIs emulate the v4 plan rate limit. If set to <code>true</code> , then API operations do not share the rate limit. This setting matches the behavior of the <code>x-ibm-gateway-emulate-v4-plan-rate-limit</code> v5 option.
<code>enable-api-logging</code>	<code>true</code> / <code>false</code> The default value is <code>true</code> .	Log all changes that the migration utility makes when rewriting APIs. These changes are logged as comments in the API YAML file.
<code>enforce-required-params</code>	<code>true</code> / <code>false</code> The default value is <code>false</code> .	When set to <code>true</code> , required parameters are enforced, which is the default behavior in API Gateway. When set to <code>false</code> , required parameters are not enforced, even if the required parameter checkbox selected.

Parameter	Valid values	Description
gatewayscript-policy	One of the following values: gatewayscript_1.0.0 , gatewayscript_2.0.0 The default value is gatewayscript_2.0.0 .	Defines which gatewayscript policy the migration utility uses when rewriting the API assembly. gatewayscript_1.0.0 : A user-defined policy created by the migration utility. This policy uses the v5 MPGW processing rule. Use gatewayscript_1.0.0 if the gatewayscript policy is dependent on v5 MPGW capabilities, such as load balancer groups or XML Manager settings. gatewayscript_2.0.0 : The migration utility rewrites the existing API to gatewayscript 2.0.0. This policy supports the header-metadata module for compatibility with the DataPower® Gateway (v5 compatible). Specifying gatewayscript_2.0.0 can cause migration errors or run-time errors. Use gatewayscript_2.0.0 if you are able to review and verify all changes made by the migration utility in the API YAML file.
if-policy	One of the following values: if_1.5.0 , switch_2.0.0 The default value is if_1.5.0 .	Defines which if policy the migration utility uses when rewriting the API assembly. if_1.5.0 : The migration utility rewrites the existing API to if 1.5.0. switch_2.0.0 : The migration utility rewrites the existing API to switch 2.0.0. Specifying switch_2.0.0 can cause migration errors or run-time errors. Use switch_2.0.0 if you are able to review and verify all changes made by the migration utility in the API YAML file.
invoke-policy	One of the following values: invoke_1.5.0 , invoke_2.0.0 The default value is invoke_1.5.0 .	Defines which invoke policy the migration utility uses when rewriting the API assembly. invoke_1.5.0 : The migration utility rewrites the existing API to invoke 1.5.0. invoke_2.0.0 : The migration utility rewrites the existing API to invoke 2.0.0. Specifying invoke_2.0.0 can cause migration errors or run-time errors. Use invoke_2.0.0 if you are able to review and verify all changes made by the migration utility in the API YAML file.
no-rename	true / false The default value is false .	Defines whether to append the porting suffix to artifact file versions or names or within artifacts. When set to false , the files are renamed using the porting suffix. When set to true , the original unpacked v5c files are overwritten with the ported API Gateway artifacts. A y/n prompt appears when using this option. To bypass this prompt, use the --accept-no-rename-deletion option.
no-retitle	true / false The default value is false .	Defines whether to append the porting suffix to the title of the migrated API, product, or OAuth provider. Set this option to true if you want the title of the migrated API, product, or OAuth provider to match the original title in the v5-compatible YAML file.
optimize-gws	true / false The default value is false .	When set to true , gatewayscript policies are modified to map v5 functions to API Gateway functions. Setting optimize-gws to true is valid only for the gatewayscript_2.0.0 policy. If it is set to true for gatewayscript_1.0.0 , an error is generated. Note: Setting optimize-gws to true can result in performance enhancements. However, the modifications to gatewayscript policies by the migration utility can cause problems. When using this setting, review all gatewayscript policy modifications after migration.
porting-suffix	A string containing alphanumeric, dash, underscore and dot characters. The default value is -apigw .	Defines the suffix to be used when renaming APIs, Products, OAuth Providers, and other ported artifacts. The suffix is appended to the version on APIs, Products and other ported artifacts that have versions, and onto the name for OAuth Providers and other ported artifacts that do not have versions. If porting-suffix is not specified then a value of -apigw will be used. The porting suffix can contain alphanumeric, dash, underscore and dot characters. Prior to AMU 10.0.5.4 the porting suffix could not be overridden and -apigw was appended to the name for all ported artifacts.
proxy-policy	One of the following values: proxy_1.5.0 , invoke_2.0.0 The default value is proxy_1.5.0 .	Defines which proxy policy the migration utility uses when rewriting the API assembly. proxy_1.5.0 : The migration utility rewrites the existing API to proxy 1.5.0 . invoke_2.0.0 : The migration utility converts the proxy policy to invoke 2.0.0. Specifying invoke_2.0.0 can cause migration errors or run-time errors. Use invoke_2.0.0 if you are able to review and verify all changes made by the migration utility in the API YAML file.
redact-policy	One of the following values: redact_1.5.0 , redact_2.0.0 The default value is redact_1.5.0 .	Defines which redact policy the migration utility uses when rewriting the API assembly. redact_1.5.0 : The migration utility rewrites the existing API to redact 1.5.0 . redact_2.0.0 : The migration utility rewrites the existing API to redact 2.0.0 . Specifying redact_2.0.0 can cause migration errors or run-time errors, including changes to what is redacted, particularly with redactions meant for JSON payloads. Use redact_2.0.0 if you are able to review and verify all changes made by the migration utility in the API YAML file.
repair-wsdl-apis	true / false The default value is true .	If an API ported from V5 is created from WSDL, the WSDL is read and checked for accuracy. If repair-wsdl-apis is set to true , then the API and WSDL are automatically repaired. For example, if the WSDL ZIP file contains unnecessary files, they are automatically removed if repair-wsdl-apis is true .
return-v5-responses	true / false The default value is false .	When set to true , OAuth-related messages are printed in the v5 response format. When set to false , OAuth-related response messages are printed in API Gateway response format.
switch-policy	switch_1.5.0 , switch_2.0.0 The default value is switch_1.5.0 .	Defines which switch policy the migration utility uses when rewriting the API assembly. switch_1.5.0 : The migration utility rewrites the existing API to switch 1.5.0. switch_2.0.0 : The migration utility rewrites the existing API to switch 2.0.0. Specifying switch_2.0.0 can cause migration errors or run-time errors. Use switch_2.0.0 if you are able to review and verify all changes made by the migration utility in the API YAML file.
use-config-file	true / false The default value is false .	Use the config.yml file that is in the same directory as the apicm binary file that is being executed. If this file is used, configuration options cannot be set using the command line. The purpose of the config file is to specify the command inputs given on this page, but in an easy-to-read file format (YAML).

Parameter	Valid values	Description
<code>v5-request-headers</code>	<code>true / false</code> The default value is <code>true</code> .	When set to <code>true</code> , the following v5-compatible request headers are set: <code>Via</code> , <code>X-Global-Transaction-ID</code> , <code>X-Client-IP</code> . When set to <code>false</code> , these request headers are not set.
<code>validate-username-token-policy</code>	<code>validate-username-token_1.0.0</code> The default value is empty.	Defines whether or not to use the <code>validate-username-token</code> v5 emulation policy when the migration utility rewrites the API assembly. When empty, the migration utility takes no action on the policy.
<code>xslt-policy</code>	One of the following values: <code>xslt_1.0.0</code> , <code>xslt_2.0.0</code> The default value is <code>xslt_1.0.0</code> .	Defines which gateway script policy the migration utility uses when rewriting the API assembly. <code>xslt_1.0.0</code> : A user-defined policy created by the migration utility. This policy uses the v5 MPGWS processing rule. Use <code>xslt_1.0.0</code> if the xslt policy is dependent on v5 MPGWS capabilities, such as load balancer groups or XML Manager settings. <code>xslt_2.0.0</code> : The migration utility rewrites the existing API to xslt 2.0.0. Specifying <code>xslt_2.0.0</code> can cause migration errors or run-time errors. Use <code>xslt_2.0.0</code> if you are able to review and verify all changes made by the migration utility in the API YAML file.

Example migration configurations

The following examples show configuration options using the `config.yml` file or the `apicm` command.

- The following example uses the `config.yml` file to specify the migration configuration.

```

deploy-policies:
- proxy_1.5.0
- redact_1.5.0
- invoke_1.5.0
- xslt_2.0.0
- gatewayscript_1.0.0
- switch_2.0.0
rewrite-apis:
  enable-api-logging: false
  proxy-policy: proxy_1.5.0
  redact-policy: redact_1.5.0
  invoke-policy: invoke_1.5.0
  xslt-policy: xslt_2.0.0
  gatewayscript-policy: gatewayscript_1.0.0
  if-policy: switch_2.0.0
  switch-policy: switch_2.0.0
  optimize-gws: true
  no-retitle: true

```

Use the `use-config-file` parameter when running the `apicm` command to specify that the `config.yml` file is used for migration configuration values.

```
./apicm archive:port-to-apigw <path> --use-config-file=true
```

- The following example uses the `apicm` command to specify the same settings as in the previous example.

```
./apicm archive:port-to-apigw <path> --deploy-policies="proxy_1.5.0,redact_1.5.0,invoke_1.5.0,xslt_2.0.0,gatewayscript_1.0.0,switch_2.0.0" --enable-api-logging=false --proxy-policy="proxy_1.5.0" --redact-policy="redact_1.5.0" --invoke-policy="invoke_1.5.0" --xslt-policy="xslt_2.0.0" --gatewayscript-policy="gatewayscript_1.0.0" --if-policy="switch_2.0.0" --switch-policy="switch_2.0.0" --optimize-gws=true --no-retitle=true
```

- The following example uses the `apicm` command to specify that the default migration configuration values are used.

```
./apicm archive:port-to-apigw <path>
```

Reviewing and correcting migrated v5 APIs

The migration utility v5 APIs to the format required by the v10 DataPower API Gateway but sometimes encounters issues that require manual intervention.

About this task

Review the following topics to understand how your v5 APIs were converted, any problems that were encountered, and whether you need to complete the conversion manually.

- [Messages generated during conversion of APIs to DataPower API Gateway](#)
Reference information about the messages generated by the migration utility command `apicm archive:port-to-apigw`.
- [Parse actions added to the API assembly after migration](#)
The migration utility command `apicm archive:port-to-apigw` adds parse actions to the API assembly depending on the policies specified in your migration configuration.
- [Fine-tuning APIs after migration](#)
After running the migration utility command `apicm archive:port-to-apigw`, review the converted APIs to determine if any of the changes made by the migration utility can be modified to enhance performance and usability.

Messages generated during conversion of APIs to DataPower® API Gateway

Reference information about the messages generated by the migration utility command `apicm archive:port-to-apigw`.

[Activity log messages](#)

[Entire API/General messages](#)

Policies

- [GatewayScript policy messages](#)
- [Invoke policy messages](#)
- [JSON-to-XML policy messages](#)
- [JWT-Generate policy messages](#)
- [LTPA-Generate policy messages](#)
- [Map policy messages](#)
- [Proxy policy messages](#)
- [Redact policy messages](#)
- [Set-Variable policy messages](#)
- [Switch policy messages](#)
- [Throw policy messages](#)
- [Validate policy messages](#)
- [WSDL messages](#)
- [XML-to-JSON policy messages](#)
- [XSLT policy messages](#)

[Migration utility messages](#)

[Miscellaneous messages](#)

Referenced objects

- [OAuth2 messages](#)
- [TLS-Profile messages](#)

Parse actions added to the API assembly after migration

The migration utility command `apicm archive:port-to-apigw` adds parse actions to the API assembly depending on the policies specified in your migration configuration.

Review the API assembly to determine if any of the parse actions or other actions added by the migration utility can be removed. For more information, see [Fine-tuning APIs after migration](#).

Parse actions are added to the assembly according to the rules in the following tables.

Table 1. Parse actions added to the API assembly after conversion of policies to 1.0.0 or 2.0.0

Policy	Where the parse action is added
If a POST , PUT , DELETE , OPTIONS , or PATCH method is found	As the first action in the assembly
invoke 2.0.0	After the invoke action, except when invoke is the last action in the assembly.
json-to-xml 2.0.0	Before the json-to-xml action
proxy 2.0.0	After the proxy action
validate 2.0.0	Before the validate action
xml-to-json 2.0.0	Before the xml-to-json action
xslt 1.0.0 or 2.0.0	Before the xslt action

Table 2. Parse actions added to the API assembly after conversion of invoke and proxy policies to 1.5.0

Policy	Where the parse action is added
invoke 1.5.0	A parse action is added after the invoke if any of the following are true: <ul style="list-style-type: none">• <code>stop-on-error</code> is specified• <code>final-action</code> is set to <code>false</code>. This is the default setting.• <code>x-ibm-gateway-optimize-invoke</code> is explicitly set to <code>false</code> The parse action is added because there can be a subsequent use of the output.
proxy 1.5.0	A parse action is added after the proxy if <code>final-action</code> is set to <code>false</code> .

Fine-tuning APIs after migration

After running the migration utility command `apicm archive:port-to-apigw`, review the converted APIs to determine if any of the changes made by the migration utility can be modified to enhance performance and usability.

If the migration utility makes any of the following changes to your APIs, consider making the suggested changes.

- If the last policy in the assembly has output that is set to a variable that is not used afterward, remove the output set or the entire policy.
- Remove any unnecessary parse actions that the migration utility adds to the assembly. For information about the parse actions added by the migration utility, see [Parse actions added to the API assembly after migration](#).
- If the assembly contains redact policies, verify that these policies are necessary. Delete any unnecessary redact policies.

Frequently asked questions about migrating APIs to API Gateway

Answers to some of the most frequently asked questions about migrating APIs to API Gateway.

- [What is port-to-apigw?](#)
- [When is it necessary to run the port-to-apigw command, and how do I use it?](#)
- [Why is the port-to-apigw command needed for migrating to v10 API Gateway?](#)
- [What is the difference between these terms: "porting to API Gateway", port-to-apigw, and "migration"?](#)
- [What is v5 policy emulation and how does it work?](#)
- [What is the purpose of the policy parameter options in port-to-apigw?](#)
- [How do the v5 emulated policies invoke_1.5.0, proxy_1.5.0, xslt_1.0.0, and gatewayscript_1.0.0 work?](#)
- [What is the difference between the 1.5.0 and 2.0.0 versions of the invoke and proxy policies, and the 1.0.0 and 2.0.0 versions of the gatewayscript policy? Why isn't the 2.0.0 version the default version?](#)
- [Should I rewrite my GatewayScript policies?](#)
- [What is the optimize-gws option?](#)
- [What is the gateway-extension.zip file and how does it work?](#)
- [How does the migration utility migrate custom policies?](#)
- [Should I set the custom-policies-scope option to global or catalog?](#)
- [Why are my custom policies not showing up in the Assembly Editor policy palette?](#)

What is `port-to-apigw`?

`port-to-apigw` is a command in the API Connect migration utility that converts all APIs, products, subscriptions, OAuth providers, and custom policies into v10 native API Gateway artifacts. These newly created artifacts live side-by-side with the original v5-compatible artifacts, and have `-apigw` appended to the end of the filename. Starting in 10.0.5.4 a porting suffix can be used to override `-apigw`.

For more information, see [Converting APIs to DataPower API Gateway](#).

When is it necessary to run the `port-to-apigw` command, and how do I use it?

Run this command if you are migrating to a v10 native API Gateway. Follow the migration steps that are appropriate for your v5 environment. For instructions on running the `port-to-apigw` command, see [Migration steps](#). For information on configuring migration options, see [Configuring migration options for DataPower API Gateway](#). For other information to help you optimize migration, see [Migration considerations](#).

Why is the `port-to-apigw` command needed for migrating to v10 API Gateway?

The required fields, properties, settings, and configuration in v10 native API Gateway differs from v5-compatible mode. The `port-to-apigw` step in the migration process is needed to convert the artifacts to conform to the changes.

What is the difference between these terms: "porting to API Gateway", `port-to-apigw`, and "migration"?

"Porting to API Gateway" describes what the `port-to-apigw` command does in the migration utility. "Migration" is the overall purpose of the migration utility. The `port-to-apigw` step is necessary for migrating to a v10 native API Gateway. Migration can be completed without the `port-to-apigw` step if you are migrating to a v10 deployment that uses only the v5-compatible DataPower Gateway.

What is v5 policy emulation and how does it work?

The v5 policy emulation layer allows both custom policies and the 1.0.x policies to run on the native v10 API Gateway. It is deployed automatically by the API Connect migration utility to the `local://isp` folder of the DataPower appliance when a custom policy is found in the v5 backup, or if you specify a 1.0.x policy in the policy option for the `port-to-apigw` command. The deployment is first staged by the `port-to-apigw` command, and pushed by the `archive:push` command. Follow the steps as outlined in [Migration steps](#).

Note that 1.5.x policies do not use v5 policy emulation, and exist only as native wrappers to the 2.0.0 implementation.

What is the purpose of the policy parameter options in `port-to-apigw`?

Input options allow you to customize your migration. If you want to use the native policies, such as `invoke_2.0.0` instead of the default `invoke_1.5.0-compatible` option, you can select the appropriate option. However, using these options can result in migration errors that require manual changes to get your APIs up and running. If you want to be up and running quickly, it is best to use the default options for the input. The default options convert APIs in a way that is most backwards compatible with v5.

For more information about the migration configuration options, see [Configuring migration options for DataPower API Gateway Configuration parameters for migrating v5 APIs](#). For explanations of errors and other messages generated during migration and suggestions for manual changes to APIs, see [Messages generated during conversion of APIs to DataPower API Gateway](#).

How do the v5 emulated policies `invoke_1.5.0`, `proxy_1.5.0`, `xslt_1.0.0`, and `gatewayscript_1.0.0` work?

When the push step of the migration utility is completed, these policies will exist on the DataPower Gateway as assembly function objects. The 1.0.x policies call the v5 policy emulation layer in order to function as if they were v5 policies. The 1.5.x policies are wrapper policies to the 2.0.0 implementation. They call the 2.0.0 policies and exist only as a compatibility layer between the original v5 policy and the new 2.0.0 policy.

What is the difference between the 1.5.0 and 2.0.0 versions of the `invoke` and `proxy` policies, and the 1.0.0 and 2.0.0 versions of the `gatewayscript` policy? Why isn't the 2.0.0 version the default version?

The 1.5.x policy implementations are native wrappers to the underlying 2.0.0 implementation. They exist only to create a compatibility layer between the v5 policy and the APiGW native policy. The `gatewayscript_1.0.0` policy uses v5 policy emulation to run as if it were on a v5 environment. The performance of the 1.0.x v5 emulation layer policies in API Gateway is the same or slightly better than the v5 and v5-compatible versions. The performance of 1.5.x policies is slightly slower than their corresponding 2.0.0 policies. You can inspect the 1.5.x wrapper implementation in Assembly Functions in DataPower Gateway.

Should I rewrite my GatewayScript policies?

Most GatewayScript policies do not require rewriting, as the native v10 API Gateway contains an enhanced compatibility layer that accepts most of the `apim` functions without modification. In most cases, GatewayScript policies require rewriting when they contain any of the following:

- inline parameters
- `require('service-metadata')` and service variables associated with it
- `require('mgmt')`
- `require('fs')`
- OAuth context variables

What is the `optimize-gws` option?

The `optimize-gws` migration option converts certain v5 gatewayscript functions to the native API Gateway implementation. This setting is optional because the compatibility layer already handles processing of original v5 gatewayscript functions. If set to true, the migration utility makes the following conversions:

- `apim.getvariable()` is converted to `context.get()`
- `apim.setvariable()` is converted to `context.set()`

- `hm.current.remove()` is converted to `context.clear()`

What is the `gateway-extension.zip` file and how does it work?

After running the `port-to-apigw` command, a `gateway-extension.zip` file is located in the `cloud/admin-org/availability-zones/default-availability-zone/gateway-services/<gateway-service>` folder. This file is automatically generated by the migration utility during the processing of the `port-to-apigw` command. It contains a `manifest.json` file, and every custom policy under the gateway service that came from the v5 backup. The `manifest.json` file contains a list of custom policies to push, and backwards-compatible policies to advertise and make available for use by APIs. These backwards-compatible policies include `invoke_1.5.0`, `proxy_1.5.0`, `xslt_1.0.0`, and `gatewayscript_1.0.0`. During the `archive:push` steps of the migration process, this zip file is pushed to the API Manager as a gateway extension.

For more information, see [Gateway extensions manifest](#).

How does the migration utility migrate custom policies?

Custom policies are migrated in multiple steps, depending on whether the `custom-policies-scope` option is set to `global` or `catalog`. Once custom policies are pushed to the DataPower Gateway, they use the v5 `policy emulation` layer that was unpacked into the `local:///isp` folder in order to run on the native v10 API Gateway.

When `port-to-apigw` is run, the following steps are performed depending on the `custom-policies-scope` setting.

- If `custom-policies-scope` is set to `global`, and there are no duplicate custom policies with differing implementations, the custom policies under each gateway service are bundled in a `gateway-extension.zip` file. This zip file contains a `manifest.json` file that describes the contents. The `gateway-extension.zip` file is copied to the `admin-org gateway-service` folder. These steps are performed automatically when `port-to-apigw` is run. Next, you must push the `admin-org` during the `archive:push` step, which pushes the `gateway-extension.zip` bundles (one per gateway) to the API Manager using an internal `APIC Toolkit gateway-extensions:create` call. The custom policies within the bundle are advertised on the API Manager. The custom policies and compatible legacy policies are shown in the policy pallet in the API Manager Assemble UI. These custom policies are also in the DataPower filesystem in the `local:///` folder. Note that if duplicate custom policies with the same name, version, and differing underlying implementation are found, the migration utility logs a message and automatically falls back to `catalog` scope to prevent the migration from failing.
- If `custom-policies-scope` is set to `catalog`, the custom policies under each gateway service are bundled in a `gateway-extension.zip` zip file. This zip file contains a `manifest.json` file that describes the contents. The `gateway-extension.zip` file is then copied to the `admin-org gateway-service` folder. The custom policy definition files under each catalog are copied into the `catalog-scoped-staging` folder, where their gateway type is set to `datapower-api-gateway`. Next, you must push the `admin-org` during the `archive:push` step, which pushes the `gateway-extension.zip` bundles (one per gateway) to the API Manager using an internal `APIC Toolkit gateway-extensions:create` call. Note that the custom policies and compatible legacy policies are not visible in the policy pallet in the UI. You can see the custom policies in the DataPower filesystem in the `local:///` folder. Before you can publish APIs that use these policies, you must push your `provider-orgs` during the `archive:push` step. The custom policy definitions as described earlier from the `catalog-scoped-staging` folder are then pushed to the API Manager automatically using an internal `APIC Toolkit policies:create` call. The APIs containing the newly migrated catalog-scoped custom policies are then published.

Note: The folders and zip files are created automatically, and you should not need to modify them. For instructions on the tasks you must perform, see [Migration steps](#).

Should I set the `custom-policies-scope` option to `global` or `catalog`?

Global scope custom policies are visible to every catalog and org on the gateway. Catalog scope custom policies are visible only to the catalog that they were migrated from. If you plan to migrate to v10 with the intention of custom policy availability the same as it is on v5, then use the default `catalog` setting.

Why are my custom policies not showing up in the Assembly Editor policy palette?

See the following troubleshooting information: [Custom policies not appearing in the policy palette after migration](#)

Logging in to a management server with an OIDC registry

Logging in with an OIDC registry involves several steps, in which you obtain a temporary token and pass it to the toolkit to ensure it can communicate securely with the management server.

Attention: If you want to log in using a non-OIDC registry, see the login steps in [Migration steps without PDUR](#) or [Migration steps with PDUR](#).

To log in, complete the following steps:

1. Enter the following command on the command line:

```
apicm login --server mgmt_endpoint_url --sso
```

where `mgmt_endpoint_url` is the endpoint URL. When you log in with a Cloud admin or Provider organization, specify one of the following URLs:

- platform API endpoint URL
- management API manager URL

These endpoint URLs are configured during the installation of API Connect, as described in [Installing the Management subsystem into a Kubernetes environment](#) and [Deploying the Management subsystem in a VMware environment](#). If you have access to the Cloud Manager user interface, you can view the configured endpoint URLs as described in [Viewing platform and UI endpoints](#), ignoring any segments at the end of the displayed URLs. If you are not sure of the endpoint URL, ask your administrator.

2. When the toolkit prompts for the context, type `provider`:

```
Context? provider
```

3. The server responds with the following message:

```
Please copy and paste the url https://mgmt_endpoint_url/auth/manager/sign-in/?from=TOOLKIT to a browser to start the authentication process.
Do you want to open the url in default browser? [y/n]: y
```

Take one of the following actions:

- If you want to use your default browser to log in to API Manager, type `y` and press Enter. The API Manager sign-in page opens in a new browser tab.

- If you don't use your default browser, type `n` and press Enter.
 - Copy the URL from the command window.
 - Open a browser, paste the URL, and press Enter.

The API Manager sign-in page displays.

4. On the API Manager sign-in page, select the OIDC registry and then log in to API Connect.
After you log in, API Connect displays the "You are authenticated!" message and provides a temporary token. Copy the token.
5. Return to the command window. Paste the token at the `API Key?` prompt and press Enter.

When the token is validated and you are successfully logged in to the toolkit, the following message displays:

```
Logged into mgmt_endpoint_url successfully
```

Migration troubleshooting

If you run migration and observe error messages in your log output, you can take troubleshooting steps, and run the migration again.

Refer to the following sections for information on resolving migration issues:

- [apicm command line help](#)
- [Log files for apicm](#)
- [Common errors](#)
- [Error messages and warnings from archive:unpack](#)
- [Error messages and warnings from archive:push](#)
- [Troubleshooting tips](#)
- [Failure to convert API definitions to API Gateway](#)
- [Custom policies not appearing in the policy palette after migration](#)

apicm command line help

You can use command line help to review usage syntax for `apicm`. For example:

```
./apicm --help
API Connect Migration Utility (AMU)
```

```
Usage:
apicm [command]
```

```
Available Commands:
archive:apistatus  Generate stats about API use
archive:automation Generate summary of the data
archive:gwstats    Generate stats about gatewayscript cmd use
archive:port-to-apigw Port artifacts to apigw
archive:push       Push unpacked content to APIConnect server
archive:unpack     Unpack a V5 management/portal archive
help              Help about any command
login             Log in to IBM API Connect
sluggify          Sluggify a name or version string
version           APIConnect migration toolkit version
```

```
Flags:
--accept-license  Accept the license for API Connect
-h, --help       Help for apicm
```

Use `"apicm [command] --help"` for more information about a command.

Example usage for more information about a command:

```
$ ./apicm archive:push --help
Push unpacked content to APIConnect server
```

```
Usage:
apicm archive:push CLOUD_MANAGER_ENDPOINT/API_MANAGER_ENDPOINT ARTIFACT_PATH
```

Locate `CLOUD_MANAGER_ENDPOINT` and `API_MANAGER_ENDPOINT` in the `apiconnect-up.yml` file used during `apic install [flags]`

```
Flags:
--apigw-only          Only attempt to push APIGW artifacts
--catalog strings    Provide a comma separated list of catalogs to push (e.g. --catalogs=cat1,cat2)
--dry-run            Prepare the working directory but do not push to server
-h, --help           help for archive:push
--no-drafts          No draft APIs or Products will be pushed to the server
-f, --overwrite      Overwrite draft APIs and products, published APIs and products,
                    and OAuth providers if they exist on the server
--saveDebugLog string[="default"] Debug logs are saved by default to logs/debug/archive-push<timestamp>debug.log
--saveErrorLog string[="default"] Error logs are saved by default to logs/archive-push<timestamp>error.log/
--saveInfoLog string[="default"] Info logs are saved by default to logs/archive-push<timestamp>info.log
                    This flag saves the file with a custom name (e.g. --saveInfoLog=myLog.txt)
```

```
Global flags:
--accept-license  Accept the license for API Connect
```

Log files for apicm

The `apicm` command creates log files that you can use to review messages produced by the `unpack` and `push` actions. All logs are saved in the `logs` directory. Unique files are saved for each run. All default log names can be overridden via the command line.

Error log

Includes all warning and error level messages.
Error logs are saved by default to:

- `logs/archive-unpackerror.log/`
- `logs/archive-pusherror.log/`
- `logs/archive-port-to-apigwerror.log/`

You can optionally give the file a different name:

```
apicm --saveErrorLog= '<filename>'
```

For example:

```
apicm archive:unpack apic33-2020-02-19-1650.tar.gz --saveErrorLog= 'unpack.txt'
```

This command creates a file `unpack.txt` in the logs directory:

```
~/example/migration-cli/logs/unpack.txt
```

Info Log

Includes all info, warning and error level messages.
Info logs are saved by default to:

- `logs/archive-unpackinfo.log/`
- `logs/archive-pushinfo.log/`
- `logs/archive-port-to-apigwinfo.log/`

You can optionally give the file a different name:

```
apicm --saveInfoLog= '<filename>'
```

For example:

```
apicm archive:unpack apic33-2020-02-19-1650.tar.gz --saveInfoLog= 'unpack_info.txt'
```

This will create a file `unpack_info.txt` in the logs directory:

```
~/example/migration-cli/logs/unpack_info.txt
```

Default path for info logs?

Debug Log

Includes all info, warning, error and debug level messages. If you encounter errors or limitations that are not documented on this page, access the debug log.
Debug logs are saved by default to:

- `logs/debug/archive-unpackdebug.log`
- `logs/debug/archive-pushdebug.log`
- `logs/debug/archive-port-to-apigwdebug.log`

Contact your IBM representative for help with using the Debug log.

Common errors

You can use the `--saveErrorLog` flag to specify that all the errors are saved in a file in `logs` folder, as described in [Saving the error log](#). Common errors:

- Long chain of error messages hours into a `push`:
This scenario typically occurs due to expiration of an authentication token. To resolve this issue, either reauthenticate and or extend the token timeout. See [Token expiration](#).
- Unable to find User Registry when migrating (pushing) a provider-org:
Occurs when the `admin-org` user registry artifacts are not shared prior to pushing provider organization artifacts. To resolve the issue, use the cloud admin account to change the visibility of the registry to `public`, and `push` the provider organization again.
- Unable to find `xxxx` when migrating (pushing) a provider organization:
Occurs when `admin-org` artifacts are not pushed prior to pushing provider organization artifacts. Ensure that all of admin org data is pushed without errors, and then `push` your provider organization again.
- Unable to create an identity provider portal-delegated-identity-provider `xxxx` because the PDUR backup data is not provided.
This error occurs if certain Portal delegated user registries are missing from the PDUR extract provided. If the missing user registries are not no longer used, they can be skipped. If they are still used, you must create a fresh backup of the PDUR.
- Unable to create product `xxxx`. The OpenAPI definition `<xyz>` requires the policy `<sample-policy>` that is not configured on the gateway-service `<sample-gateway>`:
This error occurs when the specified policy is not configured on the gateway for the catalog that contains the product. To resolve this, you must configure, on the v5 system, the specified policy on the gateway for the catalog, and re-extract the data.
- Fixing API Errors
Example:

```
WARN[2002-01-10T15:06:16-05:00] AMUUAU005W: product oauth-apis:1.0.0 requires updating before it can be pushed to a target system
Reason: product oauth-apis:1.0.0 includes content with validation errors:
Error validating API cloud/provider-orgs/onlinebanking/catalogs/oauthcatalog/products/api050-1.0.0.api.yml
- Must validate one and only one schema (oneOf) (context: (root).securityDefinitions, line: 35, col: 3)
```

```
- securityDefinitions.authorizationUrl Does not match format 'url' (context:
(root).securityDefinitions.authorizationUrl), line: 0, col:0)
```

For example, where the issue is in the API:

```
securityDefinitions:
  ClientID:
    in: query
    name: client_id
    type: apiKey
  OAuthDef:
    authorizationUrl: ""
    description: ""
```

- v5 emulated policies and custom policies not deployed to gateway. Make sure the following was done during the migration steps:
 - xml-manager is enabled in defaults of all gateways
 - Mapping of gateways was done before the porting step
 - gateway-extension.zip shows up under cloud/admin-org/availability-zones/availability-zone-default/gateway-services/<gw-name>/gateway-extension.zip
 - Ensure push of admin-org is done after all of the above is verified
- Missing Subscriptions
One of the causes for missing subscriptions could be that the products associated with those subscriptions have not been created successfully. Make sure to check the logs and see if there have been any errors during creation of those products. Please also note in these cases there would be no logs related to those missing subscriptions.

Error messages and warnings from archive:unpack

"AMUUAU000E": "The migration tool failed to initialize."

The migration utility could not access files or directories needed to establish its working environment. Ensure that file permissions allow write access to the current working directory and that the cloud directory is empty or non-existent.

"AMUUAU001E": "The {{.Type}} {{.Name}} contains content that could not be unpacked and will not be included."

The migration utility is unable to read the API Connect v5 representation of an object in the migration archive. This content is skipped and will not appear in the unpacked migration data.

"AMUUAU002F": "The archive content at {{.DocPath}} cannot be read."

The migration utility is unable to read the internal contents of a collection in the API Connect v5 migration archive. This is likely to indicate a corrupt migration archive file. The archive:unpack operation cannot be completed.

"AMUUAU003E": "The portal delegated user registry file {{.File}} cannot be unpacked and will be skipped."

The migration utility is unable to read a PDUR archive. User information associated with this user registry will not be included in the unpacked migration contents.

"AMUUAU004E": "The archive content at {{.FilePath}} cannot be read and will not be unpacked."

The migration utility is unable to read the internal contents of an individual file in the API Connect v5 migration archive. This is likely to indicate a corrupt migration archive file. This content is skipped and will not appear in the unpacked migration data.

"AMUUAU005W": "{{.Type}} {{.Name}} requires updating before it can be pushed to a target system."

The migration utility has unpacked a configuration object for which modifications are required because the content is incorrect or missing information required by a target API Connect v10 system. The unpacked file must be updated by the user before it can be published successfully to the target system.

"AMUUAU006W": "The {{.Type}} {{.Name}} has been linked to OAuth provider {{.Provider}}. Please update the security definition {{.Security}} if another OAuth Provider should be used instead."

The migration utility has added a reference from an OAuth security definition in the named API to an OAuth provider resource. The migration utility uses the authorization URL, token URL and scope information where possible to determine which OAuth provider resource is to be used with each OAuth-secured API. In some cases, the migration utility cannot accurately determine which OAuth provider is intended for use with a particular API. The OAuth provider reference can be modified if needed by editing the security definition in the unpacked API file and changing the x-ibm-oauth-provider property value to match the intended OAuth provider resource. The reference value is a relative file path that identifies an unpacked OAuth provider resource.

"AMUUAU007W": "The {{.Type}} {{.NameVersion}} could not be linked to an OAuth provider. Please update the security definition {{.Security}} to select an OAuth provider."

The migration utility has been unable to add a reference from an OAuth security definition in the named API to an OAuth provider resource.

The migration utility uses the authorization URL, token URL and scope information where possible to determine which OAuth provider resource is to be used with each OAuth-secured API. In some cases, the migration utility cannot accurately determine which OAuth provider is intended for use with a particular API. The OAuth provider reference can be added by editing the security definition in the unpacked API file and setting the x-ibm-oauth-provider property value to match the intended OAuth provider resource. The reference value is a relative file path that identifies an unpacked OAuth provider resource.

"AMUUAU008W": "{{.Type}} {{.Name}} could not be introspected for OAuth provider API references."

The migration utility has attempted to determine which OAuth providers should be configured in support of an API product, but a reference to an OAuth provider API in the product cannot be correlated with an OAuth provider resource.

"AMUUAU009W": "{{.Type}} {{.Name}} contains invalid content that cannot be unpacked."

The migration utility has encountered an entry in the API Connect v5 migration archive that contains unexpected or corrupt data. This content is skipped and will not appear in the unpacked migration data.

"AMUUAU010E": "{{.Type}} {{.Name}} cannot be unpacked to the local filesystem."

The migration utility has attempted to unpack an entry in the API Connect v5 migration archive that could not be written to the local filesystem during the archive:unpack operation. This may be the result of a local file permissions problem, an unexpectedly long file name resulting from the unpack, or a shortage of space on the local filesystem.

"AMUUAU011E": "{{.Type}} {{.Name}} contains a {{.Target}} reference that cannot be resolved."

The migration utility has encountered a SOAP API entry in the API Connect v5 archive that refers to a WSDL definition that cannot be found in the archive. This entry will not be included in the unpacked content.

"AMUUAU012W": "The {{.Type}} {{.Name}} contains a {{.Target}} reference that cannot be resolved."

The migration utility has encountered an entry in the API Connect v5 archive that refers to another entry that cannot be found in the archive. This entry will not be included in the unpacked content.

"AMUUAU013E": "Portal delegated user registry information could not be loaded for user registry {{.Name}}.
The migration utility has encountered a portal delegated user registry definition in the API Connect v5 archive for which no PDUR file name has been provided as an argument to the archive:unpack command. The user registry contents will not be included in the unpacked content."

"AMUUAU014E": "The migration tool failed to load content for the admin organization."
The migration utility was unable to find the admin organization while unpacking provider organization information in the API Connect v5 archive. This is likely to result in further errors while unpacking the provider organization contents."

"AMUUAU015E": "The migration tool failed to load provider organization contents."
The migration utility was unable to find the load provider organization contents while unpacking the API Connect v5 archive. This message indicates an internal error with the migration utility."

"AMUUAU016E": "The {{.Type}} information belonging to organization {{.Org}} could not be unpacked."
The migration utility was unable to process the provider organization contents while unpacking the API Connect v5 archive. This message indicates an internal error with the migration utility."

"AMUUAU017W": "The settings update for {{.Type}} {{.Name}} could not be completed."
The migration utility was unable to update the settings for a configuration object while unpacking the API Connect v5 archive."

"AMUUAU018E": "Apps belonging to provider organization {{.Org}} could not be converted to consumer apps."
The migration utility was unable to convert registered applications belonging to the provider organization into applications in each catalog. The conversion occurs because the representation of applications differs between API Connect v5 and API Connect v10. This message indicates an internal error in the migration utility."

"AMUUAU019W": "User registry {{.UserRegistryName}} must be migrated to a different user registry type."
The migration utility has encountered a user registry in the API Connect v5 migration archive that is not directly supported in API Connect v10. User entries can be migrated if a mapping file is used to override the user registry name and refer to another user registry that has been created in the target API Connect v10 environment."

"AMUUAU020W": "User {{.User}} in portal delegated user registry {{.Registry}} is not associated with any OIDC provider and will be skipped."
The migration utility has encountered a user entry in the API Connect v5 migration archive that belongs to a portal delegated user registry but cannot be associated with any backing identity provider. This can occur if no PDUR export file has been provided, or if the user entry information is incomplete."

"AMUUAU021W": "The TLS profile {{.Name}} includes an unsupported protocol {{.Protocol}} that will not be included in the configuration."
The migration utility has encountered a TLS profile entry in the API Connect v5 migration archive that uses a protocol that is not supported in the API Connect v10 environment. This protocol will be omitted from the unpacked entry for the TLS profile."

"AMUUAU022W": "User {{.UserName}} ({{.ID}}) will need a password reset"
The migration utility has encountered a user entry in the API Connect v5 migration archive that is associated with a local user registry but does not have a password configured. The user entry will be migrated, but the user's password reset must be reset before logging in. This may occur when user registry mappings are used to migrate users to a local user registry from some other user registry type."

"AMUUAU023W": "Uniqueness of names cannot be guaranteed when using the --sluggify-no-checksum option during unpack"
For v10.0.5.1 and earlier: The archive:unpack command has been invoked with an option that could in unusual cases result in duplicate names being used for different unpacked entries. This is likely only for names with near-identical spelling but differ only in their use of upper and lower case letters or symbols that are converted to a hyphen during the unpack process. If duplicate names appear for entries of the same type in the same context, further messages indicating problems with the duplicate will appear.
For v10.0.5.2 and later: If the migration utility is run on Windows or on Mac OS in a case-insensitive partition, additional errors may occur because of duplicate file names such as `name-1.yml` and `Name-1.yml`, which would be considered to be the same file."

"AMUUAU024E": "The stats summary could not be generated"
The migration utility was unable to produce information listing the total numbers of entries of different types that were unpacked from the API Connect v5 migration archive. This message indicates an internal error in the migration utility."

"AMUUAU025I": "The {{.Type}} {{.Name}} contains an invalid value that will be removed."
The migration utility has encountered an entry in the API Connect v5 migration archive that contains a value that is not valid for the property in which it appears. The invalid value is automatically removed. This can occur with values such as email or URL properties that contain values that do not match the required format."

"AMUUAU026I": "The {{.Type}} {{.Name}} contains an invalid value that will be replaced."
The migration utility has encountered an entry in the API Connect v5 migration archive that contains a value that is not valid for the property in which it appears. The invalid value is automatically corrected. This can occur with values such as email or URL properties that contain values that do not match the required format."

"AMUUAU027W": "The product subscription {{.Name}} cannot be pushed to the target system."
The migration utility has encountered a subscription entry in the API Connect v5 migration archive that refers to a product that is in a state that does not allow subscriptions. The subscription is removed and will not appear in the unpacked data."

"AMUUAU028E": "The migration tool failed to create default roles for {{.Type}} {{.Name}}."
The migration utility has encountered an error while attempting to create the default roles for an organization, catalog, or space. This may be the result of a local file permissions problem or a shortage of space on the local filesystem."

"AMUUAU029W": "The OAuth provider {{.OAuthProvider}} with base path {{.BasePath}} could not be unpacked."
The migration utility has failed to create an OAuth Provider with a particular base path. This can occur if there is an OAuth provider of this name with a different base path that has already been created in this provider organization. In API Connect v5, OAuth providers existed at the catalog level. In API Connect v10, they exist at the provider organization level, which can cause this issue to occur."

Error messages and warnings from archive:push

"AMUAP001E": "The {{.Type}} {{.Name}} contains nested content that cannot be published."
The migration utility is unable to read or load nested configuration content during the archive:push operation. After the migration unpack has been completed, some configuration objects are comprised of a directory with one or more nested files. For example, information about catalogs in their unpacked form consists of a directory with a name matching the catalog name, and a nested `catalog.yml` and `catalog-settings.yml` file. This message indicates that one or more of these files cannot be opened or loaded."

"AMUAP002E": "The {{.Type}} {{.Name}} cannot be published."
The migration utility is unable to read or load configuration content during the archive:push operation. This message indicates that the file representing the named configuration object is unreadable or contains unexpected content."

"AMUAP003W": "Skipping mapping file {{.Name}}."
The migration utility found a mapping file (`mapping.yml`) with invalid contents. The archive:push operation proceeds as if this mapping file were not present."

"AMUAP004E": "The file {{.Filepath}} could not be loaded and will not be published."
The migration utility is unable to read or load configuration content during the archive:push operation. This message indicates that the file representing a configuration object cannot be read or loaded."

"AMUAP005E": "The file {{.Filepath}} contains references that could not be loaded and will not be published."
The migration utility is unable to read or load configuration content during the archive:push operation. This message indicates that the file representing a configuration object cannot be read or loaded."

"AMUAP006E": "Unable to update reference to {{.Path}} in file {{.File}}."
The migration utility is unable to resolve a reference in the named file to another configuration object. This message indicates that the content in the file refers to other objects that are not found in the unpacked content, or the referenced content is not of the expected format."

"AMUAP007E": "The {{.Type}} {{.Name}} cannot be published because of a missing prerequisite."

The migration utility is unable to publish the named object because it refers to another object that cannot be found in the target API Connect system. This message indicates that the object contains an incorrect reference, or that the referenced object has not been successfully created or has been deleted from the target system.

"AMUAP008W": "The update to the {{.Type}} {{.Name}} could not be verified and may not have been successful."

The archive:push operation has attempted to update an object in the target API Connect system, but the results of the update were not received or indicated an error. This can occur in the event of a network interruption or timeout between the migration utility and the target API Connect system. If this occurs, it is safe to retry the archive:push operation; the same update will be attempted.

"AMUAP009E": "The {{.Type}} {{.Name}} was not published because it could not be checked in the target system."

The migration utility has attempted to check the existence of an object in the target API Connect system, but the check did not complete successfully. The migration utility will not overwrite objects in the target system if it cannot successfully determine whether or not a conflict will occur.

"AMUAP010W": "The creation of the {{.Type}} {{.Name}} could not be verified and may not have been successful."

The migration utility has attempted to create an object in the target API Connect system, but the results of the creation were not received or indicated an error. This can occur in the event of a network interruption or timeout between the migration utility and the target API Connect system. If this occurs, it is safe to retry the archive:push operation; if the named object was created, then it will subsequently be skipped unless the overwrite option is used.

"AMUAP011W": "The deletion of the {{.Type}} {{.Name}} could not be verified and may not have been successful."

The migration utility has attempted to delete an object in the target API Connect system, but the results of the deletion were not received or indicated an error. This can occur in the event of a network interruption or timeout between the migration utility and the target API Connect system. If this occurs, it is safe to retry the archive:push operation.

"AMUAP012W": "The ownership transfer of the {{.Type}} {{.Name}} could not be verified and may not have been successful."

The migration utility has attempted to transfer ownership of an object in the target API Connect system, but the results of the ownership transfer were not received or indicated an error. This can occur in the event of a network interruption or timeout between the migration utility and the target API Connect system. If this occurs, it is safe to retry the archive:push operation.

"AMUAP013E": "The endpoint settings for catalog {{.Name}} could not be updated."

The migration utility has attempted to update the endpoint settings for a catalog in the target API Connect system, but the update was not successful. This can occur if the catalog settings refer to a gateway service for which the configuration cannot be found in the unpacked migration contents.

"AMUAP014E": "The {{.Type}} {{.NameVersion}} could not be {{.Action}} in the target system."

The migration utility has attempted to create or modify an object in the target API Connect system, but the target system returned an error. The reason associated with the message indicates the error returned by the server.

"AMUAP015E": "The gateway extension for the gateway service {{.Name}} could not be published."

The migration utility has attempted to publish a gateway extension zip to the target API Connect system, but the target system returned an error. The reason associated with the message indicates the error returned by the server.

"AMUAP016W": "The {{.Type}} for the gateway service {{.Name}} could not be updated."

The migration utility has attempted to update the OAuth shared secret or SNI items for a gateway service in the target API Connect system, but an OAuth shared secret or SNI items were previously set. The migration utility will not overwrite an OAuth shared secret or SNI items that have already been set.

"AMUAP017W": "The product {{.NameVersion}} could not be validated and may contain errors."

The migration utility has attempted to validate a product before publishing, but the validation operation did not complete successfully. If validation errors are present in the product file, then the publish to the target API Connect system will not be successful.

"AMUAP018W": "The product {{.NameVersion}} contains API {{.APIName}} that could not be validated and may contain errors."

The migration utility has attempted to validate the APIs in a product before publishing, but the validation operation did not complete successfully. If validation errors are present in the API file, then the publish to the target API Connect system will not be successful.

"AMUAP019W": "The product {{.NameVersion}} contains validation errors: {{.Errors}}"

The migration utility has validated the content of a product before publishing, and the validation revealed errors in the product file contents. The product cannot be published to the target API Connect system unless the validation errors in the product file are corrected.

"AMUAP020W": "The product {{.NameVersion}} contains an API {{.APINameVersion}} with validation errors: {{.Errors}}"

The migration utility has validated the content of a product and its associated APIs before publishing, and the validation revealed errors in the API file contents. The product cannot be published to the target API Connect system unless the validation errors in the API file are corrected.

"AMUAP021W": "The product {{.NameVersion}} contains API content that could not be updated."

The migration utility has attempted to update the state of an API in the named product, but the metadata file (*.api.meta.yml) containing extra API attributes could not be read or contained unexpected content.

"AMUAP022W": "Subscription {{.Name}} will not be published."

The migration utility has encountered a subscription for which the subscribed product cannot be found. The subscription cannot be created in the target API Connect system.

"AMUAP023W": "The user registry {{.Name}} cannot be published."

The migration utility has encountered a user registry object that cannot be published to the target API Connect system. This can occur for user registry types that may appear in an API Connect v5 system but are not directly supported in API Connect v10. A mapping file in the unpacked user registry directory may be used to identify an alternative user registry into which user entries can be migrated.

"AMUAP024I": "Subscriptions for the product {{.NameVersion}} will not be published."

The migration utility has encountered a product that contains no subscribable APIs. This can occur when a product originating in API Connect v5 contains one or more OAuth provider APIs, and no REST or SOAP APIs. OAuth provider APIs are converted to OAuth provider resources by the archive:unpack operation, and do not appear as APIs in API Connect v10. OAuth provider instances in API Connect v10 use subscription information associated with the REST or SOAP APIs that refer to the OAuth provider in a security definition.

"AMUAP025E": "Non-fatal errors were found while running the preflight check: {{.Errors}}"

The migration utility encountered a non-fatal error during its preflight check before the start of **archive:push**. The details of any of these errors will be reported back here and the **archive:unpack** will continue.

Troubleshooting tips

- The migration utility may have issues when running across a VPN. The connection might fail if you use a corporate VPN to attempt to access an API Connect subsystem (API Manager UI) located behind an internet firewall. To work around this problem, add the IP addresses of the API Connect management server and API manager to your local /etc/hosts.
- Token expiration: By default, the **apicm** login token expires after eight hours. If the token expires while **apicm archive:push** is running, **apicm** generates errors due to lack of the necessary access permissions. In this case, log in again and re-run **apicm archive:push**. (The token is refreshed every time you log in.) Optionally, you can extend the token expiration time when logging in as the admin user:

```
apicm login --expireHours=<number_of_hours>
```

The **expireHours** flag overrides the default authentication token expiration of 8 hours. Use this flag when pushing data sets that take longer than 8 hours to complete. Note that the revised value persists after the session ends. Best practice is to revert the value to the default (8 hours) after migration.

- If your prior migration created an incorrect artifact on v10, you may have to manually remove the artifact before re-running migration. If you use the `--overwrite` flag when running additional `archive:push` commands, existing OAuth Providers, draft products, draft APIs, and published products and their APIs will be overwritten.
- When you re-run migration, the utility will not recreate any artifacts that already exist on v10 unless the `--overwrite` flag is used for OAuth Providers, draft products, draft APIs, and published products and their APIs. If a collision occurs for other artifacts or if the `--overwrite` flag was not used, the older artifact takes priority. Also, the utility will not delete any artifacts from v10. Thus, you can re-run migration to cover any gaps in artifact creation on v10 without affecting existing artifacts.
- The steps you need to take depend on whether you need to change the v5 export data. For example, if you need to publish or replace a v5 API, you need to export the v5 data and run `unpack` again. If you do not need to change the v5 data, you can skip those steps.

Example 1: Re-using the v5 export data

For example, your v5 export data is accurate and current, but you migrated a consumer-org that is no longer needed on v10, and you mapped a catalog to an incorrect name. Complete the following steps:

1. Delete the incorrect artifacts from the v10 deployment `/cloud` directory. For this example, delete the unneeded consumer-org and the incorrectly named catalog.
2. Edit the contents of the `/cloud` directory as needed. For example, remove the unneeded consumer-org entry.
3. Edit your mapping entries as needed. For example, remap the catalog name.
4. Run `apicm archive:push <server> <project_directory>`.

Example 2: Revising v5 export data

In this example your v5 export data is not current because you want to publish or replace a v5 API and then migrate it to v10. Complete the following steps:

1. On v5, recreate the export data: `config dbextract sftp <host_name> user <user_name> file <path/name>`
2. On v10, delete the `/cloud` directory.
3. Run `apicm archive:unpack` to create the v10 hierarchy in `/cloud` directory.
4. Recreate any edits you made to the `/cloud` directory.
5. Recreate any mappings.
6. Run `apicm archive:push <server> <project_directory>`.

Failure to convert API definitions to API Gateway

When you change gateway types from DataPower Gateway (v5-compatible) to DataPower API Gateway, you use `./apicm archive:port-to-apigw <path>` to convert API definitions to API Gateway format. The conversion requires Node version 18.17.1 or greater. If Node is not installed, this error is displayed:

```
ERRO[0000] Error getting node version: exec: "node": executable file not found in $PATH
```

If an unsupported version of Node is installed, an error is displayed as in the following example:

```
ERRO[0000] Node version installed must be >=18.17.1. Currently installed version: 4.9.1. Use the --allow-older-node flag to allow the use of an older node 18 version at the users own risk.
```

To resolve the error, install Node and re-run the command. If you are on an older version of node 18 you can re-run the command using the `--allow-older-node` flag but you do this at your own risk.

To view the conversion step, see [Migration steps without PDUR](#) or [Migration steps with PDUR](#). For information about the conversion, see [Reviewing APIs converted to DataPower API Gateway](#).

Custom policies not appearing in the policy palette after migration

If you are migrating to v10.0.1.3 or earlier, or v10.0.2.0 or earlier, follow these steps to have custom policies appear in the policy palette in the API Manager Assembly UI.

1. Open `gateway-extension.zip` in the `cloud/admin-org/.../<gateway-service>` folder that is generated in `admin-org` after `port-to-apigw` is run.
2. Open the `manifest.json` file.
3. Edit the `manifest.json` file to specify the implementation files for the custom policies to push to the sandbox catalog. Add entries for the sandbox catalog for all provider-orgs using the format `<provider-org>_<catalog-name>`. The filename can remain the same for each respective provider-org, as it will be the same custom policy. Ensure that the actual file that the filename is pointing to in the `.zip` file exists. The same filename can be re-used multiple times in the `manifest.json` to add the sandbox entries for the custom policy. Only the `manifest.json` must be edited. You do not need to edit any of the `.zip` files in the `gateway-extension.zip` package.

4. Save the `manifest.json` file and zip it.
5. If the `archive:push` was already performed before step 1, then delete the `gateway-extension.zip` file from the gateway using the API Connect toolkit command `apic gateway-extensions:delete`.
6. Push `admin-org`, which pushes the updated `gateway-extension.zip` file.
7. Follow these steps to ensure that the API Manager advertises the custom policy that is being pushed.
 - a. In each of the provider orgs in the cloud folder, navigate to the `catalog-scoped-staging` folder and log in to the API Connect toolkit as a provider-org user.
 - b. Run this command on every `.zip` file that you want to appear in the policy palette in the v10 UI:

```
apic policies:create -c sandbox --configured-gateway-service -o -s $API_SERVER_URL --scope catalog .zip
```

The `gateway-service` name is under Topology. `$API_SERVER_URL` is the same one you use to log in to the API Connect toolkit. The custom policy zip filename will appear in the `catalog-scoped-staging` folder.

8. Navigate to the API Manager UI and check the assembly policy palette. The custom policies should now appear in the list. You do not need to push the provider orgs.

If you are migrating to v10.0.1.4 or later, v10.0.3.0 or later, follow these steps to have custom policies appear in the policy palette in the API Manager Assembly UI.

1. After the migration of `admin-org` and `provider-orgs` has completed, navigate to the API Manager UI.
2. While editing a draft API, click the gear icon next to the offline button. A pop-up should appear.
3. Select the `target-catalog` to publish the draft API to. The `target-catalog` must already have the custom policies pushed to it.
4. Select the Application to use for the API.

5. Click Save.
6. Navigate to the Assembly policy palette. The custom policies for that specific target-catalog should now appear. For more details about these steps, see [Specifying the testing preferences for an API](#).

When artifacts such as provider org, space or catalog are mapped, any warning or error messages for artifacts in subdirectories underneath those may have the location incorrect. The location will include the mapped artifact location instead of the original location.

Migrating a Version 5 deployment

Use the Migration Utility to automate the migration of a v5 deployment to a v10 deployment.

Supported versions

- You can migrate API Connect Version 5.0.8.7 or later point release to API Connect Version 10.0.2.0 or later point release. If your Version 5 system is older than the minimum, you must first upgrade it to the minimum version.
- You can migrate API Connect Version 5.0.8.7 or later point release to a Cloud Pak for Integration version 2021.4 or later. The version of Cloud Pak for Integration must use API Connect Version 10.0.4.0 or later.
- The migration utility version must match the API Connect and DataPower version. Observe the following Version 10 API Connect requirements:
 - For Kubernetes, the installation operator version must match the API Connect version.
 - For VMware, the `apicup` utility version must match the API Connect version.

Obtaining the API Connect Migration Utility

Download the API Connect Migration Utility (AMU) from IBM Fix Central. For a link to the latest files on Fix Central, see [What's new in the latest release](#).

- [Migration overview](#)
The migration process provides tools to extract v5 data, arrange it for v10 formats, and load it onto a v10 system.
- [Migration steps](#)
Plan the migration, install a new v10 deployment, and move data from v5 to v10.
- [Converting APIs to DataPower API Gateway](#)
- [Migration mapping files](#)
You can use mapping files to map the name of a v5 object to a corresponding v10 object that has a different name.
- [Migrating OAuth providers and OAuth secured APIs](#)
You can migrate v5 OAuth providers and OAuth secured APIs to your v10 API Connect deployment.
- [Migrating custom policies](#)
The migration utility migrates API Connect Version 5 user-defined custom policies to API Connect Version 10. After migration, the v5 custom policies run on a v5-compatible DataPower Gateway or the DataPower API Gateway. Custom policies can be migrated to API Gateway with limitations.
- [Migrating gateway extensions](#)
Use the appropriate procedure for migrating extensions to either a v10 DataPower® Gateway (v5 compatible) or a v10 DataPower API Gateway.
- [Migration considerations](#)
You can optimize the migration process by reviewing the migration considerations and taking any actions that apply to your deployment.
- [Migration troubleshooting](#)
If you run migration and observe error messages in your log output, you can take troubleshooting steps, and run the migration again.

Migration overview

The migration process provides tools to extract v5 data, arrange it for v10 formats, and load it onto a v10 system.

Important: The migration utility allows migration from v5 to a v10 v5-compatible gateway or v10 API Gateway. References to conversion of v5-compatible artifacts to v10 API Gateway refer to the v5-compatible artifacts that were initially from a v5 environment and were unpacked by the migration utility.

The migration model is centered around using tooling to bring the contents of Version 5 catalogs onto a Version 10 deployment that is set up to import them. The tooling migrates provider organization configuration information, including APIs, Catalogs, Products, Consumers, subscriptions, user registries and more.

The migration utility also supports migrating your billing data from v5 to v10:

- All Stripe billing integrations in v5 are migrated to v10 and applied automatically to all Catalogs.
- All products that use paid plans will have the plans migrated.
- Consumer payment methods that exist in the v5 Portal will be automatically created in v10.

After the migration is complete, you must configure the Stripe integrations in the Developer Portal sites; see [Configuring Stripe in the Developer Portal](#) for detailed instructions. For more information about the billing feature in v10, see [Monetizing your Products](#).

You use a v5 extract command to gather the v5 data, and then use a migration unpack command to convert the data into Version 10 format. This extraction will generate a `c1oud` directory where your v5 data will be represented as YAML files. By viewing the contents of the `c1oud` folder, you can see which structures, such as provider organizations, are needed to support your catalog.

The model recognizes that not all v5 data is applicable to a new Version 10 deployment. To optimize your v10 deployment, you may use a text editor to remove unneeded or obsolete v5 configurations from the converted file. When you install and configure your v10 management server, you manually create those structures. You can then use migration tooling to load the v5 data into the prepared v10 system.

During the migration process you can keep your existing v5 system active while you install and configure a parallel (side-by-side) v10 system. You must have system administrator privileges to complete a migration. You should be familiar with IBM API Connect Version 5 architecture and requirements, and with the deployment configuration that you want to migrate.

The migration process supports migrating v5 data to both v10 DataPower Gateway (v5-compatible) and v10 DataPower® API Gateway.

The following sections in this topic review each step in the migration process. See also [Migration considerations](#).

Migration steps

Plan the migration, install a new v10 deployment, and move data from v5 to v10.

About this task

At the highest level, the migration steps are:

- Plan your migration.
- Install and configure a v10 system to best meet current needs and integrate v5 data.
- Backup your v5 data and migrate the data to your new v10 deployment.

Procedure

Complete the following steps:

1. Plan your migration:
 - Read [Migration overview](#).
 - Ensure you have supported versions of the software. See [Supported versions](#).
 - Plan your new v10 deployment. When moving from v5 to v10, use the opportunity to review your topology, user registries, provider organizations, etc. You might want to look at the existing data on the v5 system, and do some cleanup before running the extract. This will help ensure only objects that are required in v10 are migrated over.
 - If you are migrating to DataPower API Gateway, migrate a proof-of-concept deployment before migrating your entire v5 backup. Migrate 10 to 20 of your most commonly used v5 APIs and test them to determine which migration utility settings you may need to convert your APIs to DataPower API Gateway.
2. Install and configure a v10 system.

Install and deploy v10 with usage of the incoming v5 data in mind. Before you push your v5 data to v10, your v10 system must be up and running, with gateway services, provider organizations, user registries, etc, ready to receive the data.

- For the v10 deployment, install the API Connect subsystems. Use the normal installation and configuration processes as shown in the documentation:
 - [Installing and maintaining IBM API Connect](#)
 - [Configuring and managing your server environment](#)

Important:

Verify that you completed the following steps as part of deploying your v10 environment:

- Ensure that you create the same number of API Managers as on the v5 deployment.
- Use the Cloud Manager to configure topology. See [Defining your topology](#).
- If your v10 topology uses identical names and quantities of gateway services as v5, the migration utility automatically maps configuration information. If your v10 configuration uses different names or a different number, you must manually map the v5 gateway services to v10 gateway services. You will be guided to map gateway services in a later step. To review the mapping requirement, see [Mapping a gateway service](#).
- Review Gateway types and configurations:

API Connect v10 supports two types of gateway services: DataPower Gateway (v5-compatible) and DataPower API Gateway. The DataPower API Gateway type is not supported for API Connect v5. When you register your gateway service on your v10 system, you must select the gateway service type. If you select API Gateway, your migration process must include a step to convert (port) the v5 API Definitions from the DataPower Gateway (v5-compatible) format to the DataPower API Gateway format. The migration utility provides an option to do the porting. Command line instructions are provided in a later step.

If migrating to datapower-api-gateway, confirm xml-manager is enabled in default domain of all gateways.

To review gateway types and configuration, see [API Connect gateway types](#) and [Registering a gateway service](#).

- Create the provider organizations. If your v10 configuration for provider organizations uses different names or a different quantity, you must manually map the v5 provider organizations to v10 provider organizations. You will be guided to map provider organizations in a later step. To review the mapping requirement, see [Mapping a provider organization](#).
 - When migrating catalog spaces in an provider organization that uses a local user registry (LUR), ensure that the LUR is configured with the visibility setting of either **Public** or **Custom**. If set to Custom, ensure that the provider organization is selected. For information on configuring visibility, see [Setting visibility for a user registry](#).
 - Configure user registries such as LDAP, as needed to match the user registries used for v5 users.
 - The migration process includes migration of users. Configuration of users into API Connect includes sending email to users to set up their accounts. The recommended way to do this during an automated migration is to set up a temporary SMTP server. One open source option for SMTP is MailHog. See <https://github.com/mailhog/MailHog>. When the migration process is complete, be sure to replace your temporary SMTP server with the permanent SMTP server for your API Connect v10 deployment.
 - Create the Portal services that you want to use with the Catalogs that you plan to migrate. You will configure the catalogs to the Portals after the migration data is pushed to the v10 system.
3. Backup your v5 data and migrate the data to your new v10 deployment.
 - a. Download the migration utility. See [Obtaining the API Connect Migration Utility](#).
 - b. Determine if your v5 system contain catalogs that user a Portal Delegated User Registry (PDUR).

Use of the PDUR was optional in v5. Migration of systems that use PDUR is very similar to migration of systems that do not use PDUR, however a few additional steps are needed. If you are not sure if your v5 catalogs use PDUR, check your v5 registry settings. See [Selecting the Portal Delegated User Registry](#).
 - c. Continue with instructions that apply to your deployment:
 - [Migration steps with PDUR](#)
 - [Migration steps without PDUR](#)

- [Migration steps with PDUR](#)

Complete the following steps to migrate from v5 to v10 when your v5 catalogs use Portal Delegated User Registries (PDUR).

- [Migration steps without PDUR](#)
Complete the following steps to migrate from v5 to v10 when your v5 catalogs do not use Portal Delegated User Registries (PDUR).
- [Logging in to a management server with an OIDC registry](#)
Logging in with an OIDC registry involves several steps, in which you obtain a temporary token and pass it to the toolkit to ensure it can communicate securely with the management server.

Migration steps with PDUR

Complete the following steps to migrate from v5 to v10 when your v5 catalogs use Portal Delegated User Registries (PDUR).

Before you begin

Important: If your v5 deployment does **not** use Portal Delegated User Registry (PDUR), do not use these instructions. See [Migration steps without PDUR](#).

- Ensure you planned your migration, obtained the migration utility, and installed a v10 deployment, as described in [Migration steps](#)
- Review [Migration considerations](#).
- Ensure you have system administrator privileges.
- To use `apicm` to convert the API definitions, you must have Node version 18.17.1 (or greater) installed.

About this task

Tip:

To achieve a smooth and successful migration, we recommend these best practices:

- Before you push your v5 data to v10, your v10 system must be up and running, with gateway services, provider organizations, user registries, etc, ready to receive the data.
- Plan to do several test iterations of migration. You can use the `--dry-run` option to do iterations of the migration push, to test for any errors or warnings to address before the actual data push.
- Use the `apicm <command> --help` command to review the command options and their purpose. To see the list of commands you can run: `apicm help`.
- Complete all the steps in this topic in the order listed, unless the step is marked optional and does not apply to your scenario. You can push provider orgs as many times as needed.

The migration of v5 catalogs that use PDUR to a v10 deployment contains all of the steps for migration a non-PDUR v5 deployment to a v10 deployment (as described in [Migration steps without PDUR](#)) plus additional steps that are needed to migrate Portal Delegated User Registry data.

In IBM API Connect Version 5, when Portal Delegated User Registry (PDUR) was selected for a Catalog, the user management was delegated from the management server to the Developer Portal. The site administrator within the Portal site then performed the configuration. The management server recorded that the delegation had taken place, and stored a record of the users which existed in the catalog. However, the management service maintained no information about the configuration of the specific registries or how users mapped to them.

PDUR in IBM API Connect v5 was highly configurable. A single Portal site (catalog) could be configured to use multiple different registry types or different registries of the same type. Details of this configuration, and which users map to which registry, is all stored in the Portal.

However, IBM API Connect Version 10 and later supports new registries, and no longer needs the PDUR. Instead, functionally equivalent registries can be configured in v10 and used. To migrate the PDUR user registries and users from v5 to v10, we must gather configuration information from both the management server and the Portal node.

PDUR supported 4 different types of user registries. The following table shows the mapping of these types to the v10 equivalents:

Table 1. Mapping of Registry types

v5 PDUR Registry	v10 Equivalent Registry
LDAP	LDAP
OIDC	OIDC
HybridAuth	OIDC
Local User Registry (LUR)	Local User Registry (LUR)

In this scenario, if you want to migrate users from an IBM API Connect V5 Developer Portal to an IBM API Connect v10 Developer Portal, you must export the user information that is stored in the Developer Portal local database. This information can then be used by the migration tooling to migrate the PDUR user registry. You complete this action after you have exported the v5 configuration database for the management server.

Procedure

1. Log in to the command line interface of the v5 appliance. Use the following command to create a tar.gz export of the v5 configuration database and place it on an `sftp` server:

```
config dbextract sftp <host_name> user <user_name> file <path/name>
```

- Specify a name of your choice for the compressed file to contain the dbextract. For example, `v5mgmt_data.tgz`.
- If you are not familiar with the v5 appliance CLI, see [The Command Line Interface](#). For information on `config dbextract`, see [Configuration commands](#).

2. Log in to Developer Portal CLI to export Portal Delegated User Registry user information for all of the sites on a Developer Portal server. Use the following command to create a `*.tgz` file:

```
pdur_user_export
```

The status of the export is displayed. For example:

```

admin@apimxxx123:~$ pdur_user_export
portal123.company.com/api-provider/production: Exporting pdur users
portal123.company.com/api-provider/preprod: Exporting pdur users
portal123.company.com/api-provider/uat: Exporting pdur users
portal123.company.com/api-provider/development: Skipping as not a pdur enabled site
portal123.company.com/api-provider/sandbox: Skipping as not a pdur enabled site
~/pdurexport ~
5c20b903e4b0485cbbd81775.5c20bf43e4b0485cbbd81795.json
5c20b903e4b0485cbbd81775.5c2cddbae4b0485cbbd818a0.json
5c20b903e4b0485cbbd81775.5c335211e4b0485cbbd8196e.json
~
Output can be found in /home/admin/pdurexport.tgz

```

Results:

- When the command completes, you can see the output `*pdurexport.tgz` in `/home/admin/pdurexport.tgz`.
- This compressed file contains all of the JSON user-mapping files for that server. Each .json file contains a list of all of the PDUR user information that is stored in the Developer Portal local database. The naming convention used for the individual user-mapping files is `<provider_org_id>.<catalog_id>.json`.
- The user information for all of the user registration methods that are used in the specific Developer Portal are included in the same file. For example, local user registry, LDAP user registry, third-party authentication, OpenID Connect, and any custom user registration method.
- If a site doesn't contain any PDUR data, that site is skipped. The example output shows that there was one provider organization on the server, and that organization had five Developer Portal sites, three of which contained PDUR user information. The PDUR user information for all of the sites on the server is exported into a `pdurexport.tgz` file.

3. Optional: Setup a temporary SMTP server.

During migration, if subscription approval is needed to enable an application to subscribe to a plan, notifications are sent to all users, within the organization, who have subscription approval permission. In some cases, this results in a large number of emails. To avoid the sending of these emails, set up a temporary email server prior to running the migration, and use it to collect all notifications rather than sending them to users' email accounts.

If subscription approvals are not needed during your migration, you can skip this step.

4. Optional: If necessary, set up a temporary LDAP server. This can be needed if you are temporarily behind a firewall.

a. Install a server:

```

git clone https://github.com/veo-labs/ldap-server-mock
npm install

```

b. Creating the configuration files:

- `ldap-server-mock-conf.json`

```

{
  "port": 3004,
  "userLoginAttribute": "cn",
  "searchBase": "dc=test",
  "searchFilter": "(&(objectclass=person)(cn={username}))"
}

```

- `users.json`

```

[
  {
    "dn": "cn=user,dc=test",
    "cn": "user-login"
  }
]

```

c. Run the LDAP server:

```

node server.js --conf=./ldap-server-mock-conf.json --database=./users.json

```

d. Modifying a current LDAP provider.

The following file is a dummy user registry file that maps to a read-only test server. The fields in bold font (`admin_dn` and `endpoint`) have been changed to fit the dummy server configuration. Change these on all user registries that need to be fixed.

```

api_version: 2.0.0
configuration:
  admin_dn: cn=user,dc=test
  admin_password: xxxx
  authenticated_bind: "true"
  authentication_method: search_dn
  bind_prefix: ""
  bind_suffix: ""
  protocol_version: "3"
  search_dn_base: ""
  search_dn_filter_prefix: (examplelegacyuid=
  search_dn_filter_suffix: )
  search_dn_scope: sub
endpoint:
  endpoint: ldap://172.20.20.11:3004
  tls_client_profile_url: idurl://..\..\..\admin-org\tls-client-profiles\tls-profile-for-ldap-1.0.0.yml
integration_url: file://..\..\..\integrations\user_registry\ldap.yml
metadata:
  v5_id: xxxxxx40cf2fc09a9ba0d6c
  v5_scope: '["api"]'
  v5_type: ldap
name: xxxxxxxxxxxxxxxx
title: xxxxxxxxxxxxxxxxxxxxxxxx
type: user_registry

```

5. Unpack both the v5 management data backup and the PDUR tar file.

Important: Migration on Windows 10

Windows has path length limits that can interfere with large extracts. It is recommended that you keep the initial path for migration on Windows as short as possible. The AMU must be run in a command window with administrative privileges on Windows. Running from a Linux VM on your Windows system may be a

better option for doing migration.

- Copy the v5 extract tar file (`<v5mgmt_data>.tgz`) and the PDUR extract file (`pdurexport.tgz`) from the sftp server to the location where you have v10 migration utility installed.
- Run `apicm archive:unpack` command from the directory where you downloaded the v5 extract file. When you invoke `apicm` for the first time, you are prompted to accept the license. Accept the license to continue.

You can optionally limit the unpack to only specific provider orgs. See [Table 2](#).

```
./apicm archive:unpack <v5_extract_file>.tgz pdurexport.tgz
```

Table 2. `apicm archive:unpack` command options

Parameters	Description
<code><v5_extract_file></code>	Can be compressed file (.tgz, tar.gz) or uncompressed file (.tar). Use any file name of your choosing. Example: <code>v5_DBextract_data.tgz</code>
<code>pdurexport.tgz</code>	The Portal Delegated User Registry export file generated by the <code>pdur_user_export</code> utility. Example: <code>pdurexport.tgz</code>
<code>--provider-orgs</code>	One or more provider orgs to unpack. This flag is optional. Use it when you want to unpack only specific provider orgs rather than the entire archive. Example: <code>./apicm archive:unpack v5_DBextract_data.tgz pdurexport.tgz --provider-orgs=<pOrg1_name></code> Use a comma-separated list to specify multiple provider orgs: <code>./apicm archive:unpack v5_archive_data.tgz pdurexport.tgz --provider-orgs=<pOrg1_name>,<pOrg2_name>,<pOrg3_name></code>
<code>--use-old-sluggify</code>	Unpack the V5 configuration data onto the local filesystem using the version 10.0.5.1 conversion rules for slugging element names and version numbers. For more information about the conversion rules, see Parity between v5 and v10 .

Note:

- You can avoid the license prompt by adding `--accept-license` to the command line.
- The unpack command ensures that catalog names and API properties contain only valid characters. Any invalid characters are mapped to '-'. See [Parity between v5 and v10](#).
- The unpack command issues warning messages for APIs and Products that fail validation against the OpenAPI specification. See the limitations section in [Migration considerations](#).
- For troubleshooting tips, see [Migration troubleshooting](#)
- The unpack command might return error messages related to PDUR, such as:

```
ERRO[3414] PDUR backup data for IDP PortalDelegatedIdentityProvider-123b81fbe4b06de411580e2e (123b81fbe4b06de411580e2f) not provided: not found
ERRO[3414] PDUR backup data for IDP PortalDelegatedIdentityProvider-123b820de4b06de411580e32 (123b820ee4b06de411580e33) not provided: not found
```

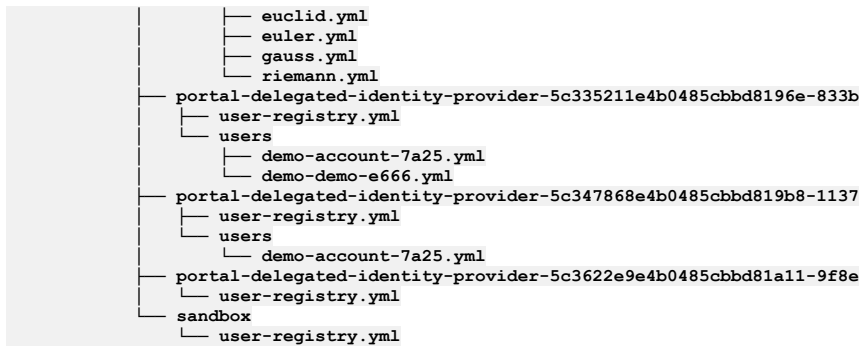
The error means that when the migration utility unpacked `pdurexport.tgz`, it could not find all the specified Portal user registries. To resolve the errors, re-run `pdur_user_export` and re-run the unpack..

When the command completes, the data is unpacked to the `/c1oud` directory. The `/c1oud` directory is located in the same directory as `apicm`, and populated with yaml files representing the configuration objects in the export. The unpack processes produce output in the format required for v10 configuration.

Note: The unpack directory must be named `/c1oud`. Do not rename the directory.

Example output in `/c1oud` for provider-org:

```
├── provider-orgs
│   ├── pOrg1
│   │   ├── catalogs
│   │   │   ├── catalog1
│   │   │   └── ...
│   │   └── user-registries
│   │       ├── pdur-ldap-5c38c102e4b0485cbbd81a84
│   │       │   ├── mapping.yml
│   │       │   ├── user-registry.yml
│   │       │   └── users
│   │       │       ├── gauss.yml
│   │       │       └── riemann.yml
│   │       ├── git-hub-5c38c102e4b0485cbbd81a84-bbfe
│   │       │   ├── user-registry.yml
│   │       │   └── users
│   │       │       └── apictestgithub-apictestgithub-c646.yml
│   │       ├── google-5c38c102e4b0485cbbd81a84
│   │       │   ├── user-registry.yml
│   │       │   └── users
│   │       │       └── demo-account-7a25.yml
│   │       ├── lur-5c38c102e4b0485cbbd81a84
│   │       │   ├── user-registry.yml
│   │       │   └── users
│   │       │       └── lur-user-1.yml
│   │       ├── portal-delegated-identity-provider-5c20b903e4b0485cbbd81781-7293
│   │       │   ├── user-registry.yml
│   │       │   └── users
│   │       │       ├── anotheruser.yml
│   │       │       ├── onemore.yml
│   │       │       └── pduruser.yml
│   │       ├── portal-delegated-identity-provider-5c20bf43e4b0485cbbd81795-954
│   │       │   ├── user-registry.yml
│   │       │   └── users
│   │       │       ├── anotheruser.yml
│   │       │       ├── onemore.yml
│   │       │       └── pduruser.yml
│   │       ├── portal-delegated-identity-provider-5c2cddb4e4b0485cbbd818a0-986d
│   │       │   ├── user-registry.yml
│   │       │   └── users
│   │       │       └── einstein.yml
```



6. If the v5 Developer Portal used LDAP, HybridAuth, or OIDC (such as social login through Google, GitHub, or Slack) user registries, you must manually configure them in v10 before using `apicm archive:push` to migrate PDUR. See the following instructions for your registry type:

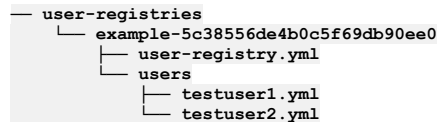
- LDAP

- Manually configure LDAP

The user registries which are required in the target v10 system are described in the `user-registry.yml` file. Where possible the configuration for the v10 registries is complete in these files. However, for LDAP the configuration of the registry is not possible due to differences in the way the v5 Portal (Drupal) and v10 API Manager (`apim`) specify user registry configuration.

For LDAP with v10, create a user registry manually, using the same LDAP server and configuration as used with v5. Where it is possible, the v5 configuration is provided as helpful information in the `metadata.v5_config` property. To view the v10 configuration instructions, see [Configuring an LDAP user registry](#).

For example: An `ldap.tar.gz` file, when unpacked, creates the following directory hierarchy, for an LDAP directory `example`:



You can create an LDAP user registry for the PDUR by either modifying `user-registry.yml` or by creating the registry in the UI:

- Modify the `user-registries/example-5c38556de4b0c5f69db90ee0/user-registry.yml` file directly and add the required LDAP fields based on the extracted info provided in the metadata. Remove the metadata after you are done. The modification of `user-registry.yml` is preferred over using the UI. This method enables any APIs that depend on the user registry to be satisfied when the migration command `apicm archive:push` is run in Step [10](#).
- Create the registry in the UI. The creation of the registry in the UI requires more manual setup of the organization and catalog. In addition, you must provide a mapping file to map from the `example-5c38556de4b0c5f69db90ee0` to the ID of the registry that was created manually.

- Create a mapping file entry for the LDAP user registry.

In the `/cloud` directory, go to `provider-org/<provider-org>/user-registry/<PDUR_LDAP_User_registry>` and create a mapping file to tell migration utility the name of the migration endpoint for the LDAP user registry used with v5 PDUR:

```

user-registry:
  name: <pdur-ldap-v10-name>

```

To determine which name to use for the v10 LDAP registry, go to the management user interface, click on **Resources** -> **User Registries** then use **Name** not **Title**.

For more information on how to create mapping files, see [Migration mapping files](#).

- OIDC

- Manually configure OIDC

- Note that v10 supports OAuth 2.0 but does not support OAuth 1.0. Deployments, such as Twitter, which use OAuth 1.0 with APIC version 5, can not use OAuth 1.0 when migrating to v10.
- Automatic migration is not supported for the OIDC user registries which you use in your v5 Portal. Thus if you are using OIDC OAuth 2.0, e.g., Google, GitHub, or Slack, please complete the following manual steps.
- The OIDC data from the Portal Delegated User registry does not contain the `client_id` and `client_secret` that are needed to migrate a user registry automatically. You must obtain the data and update the user registry manually if you are using `client_id` and `client_secret`.

On the Version 10 system:

- Edit `user-registry.yml`. For example, for GitHub: `cloud/provider-orgs/test-user1-org1/user-registries/git-hub-5c5xyz6ee4b0c5f69db94xxxx-a339/user-registry.yml`

```

api_version: 2.0.0
configuration:
  client_id: ""
  client_secret: ""
  authorization_endpoint: https://github.com/login/oauth/authorize
  provider_type: github
  token_endpoint:
    endpoint: https://github.com/login/oauth/access_token
integration_url: file://../../../../integrations/user_registry/oidc.yml
metadata:

```

```
v5_id: 5c5xyz6ee4b0c5f69db94xxxx
name: git-hub-5c5xyz6ee4b0c5f69db94xxxx-a339
registry_type: oidc
title: GitHub
type: user_registry
```

- Insert the values for `client_id` and `client_secret`. Once the client id and client secrets have been updated in the `user-registry.yml` file just described, the `apicm archive:push` can be successfully executed. Upon creation and configuration of the Developer Portal, you can log in by using, as in this example, an account such as a GitHub account.

To verify that the manual configuration succeeded, view the log file for entries for the users after you run `apim archive:push` in Step [10](#).

- Example success message:

```
INFO[0012] User registry git-hub-5c5xyz6ee4b0c5f69db94xxxx-a339 created
INFO[0013] User test-user1-4d48 created
```

- Example error message:

```
ERRO[0010] unable to create User registry git-hub-5c5xyz6ee4b0c5f69db94xxxx-a339
ERRO[0010] git-hub-5c5xyz6ee4b0c5f69db94xxxx-a339: The properties client_id, client_secret are required.
```

- Create a mapping file entry for the OIDC user registry.

In the `/cloud` directory, go to `provider-org/<provider-org>/user-registry/<PDUR_OIDC_User_registry>` and create a mapping file to tell migration utility the name of the migration endpoint for the OIDC user registry used with v5 PDUR:

```
user-registry:
  name: <pdur-OIDC-v10-name>
```

For more information on how to create mapping files, see [Migration mapping files](#).

7. Optional: Remove unneeded configuration objects.

Inspect the set of configuration objects and their attributes, and remove the ones that are not needed. In many cases, some of the configuration objects present in the v5 system are not needed in the new v10 environment. For example, there is no value in preserving inactive users, obsolete apps, obsolete consumer organizations, and so on.

When deleting objects, make sure that you do not delete objects that have other dependent objects, such as products with subscriptions on them or users that are active. If you delete a consumer organization make sure you also delete all apps and subscriptions associated with that consumer org. Be aware of any groups that include these consumer orgs or visibility settings that include the consumer orgs.

8. Map v5 object names to v10 object names.

Most deployments adjust the names or quantities of gateway services, provider organizations, and user registries when setting up a new environment for v10. Be sure to specify any changed v10 destinations for the v5 data in a migration mapping file so that the migration utility can automatically move the data.

Complete the following, as needed. You can skip the steps for any objects that are unchanged from v5 to v10.

- [What is a mapping file?](#)
- [Mapping a gateway service](#)
- [Mapping a provider organization](#)

9. Optional: If your v10 deployment uses a DataPower API Gateway, convert the v5 API definitions to v10 DataPower® API Gateway.

To convert API definitions using the default migration configuration, use the following syntax.

```
./apicm archive:port-to-apigw <path>
```

Important: Ensure that the DataPower Gateway version that you are using with your API Connect v10 deployment is compatible with the version of the migration utility that you are using. For details, see the version compatibility information in [Configuring and managing your server environment](#).

Important: If you are running the migration utility in Windows, Administrator privileges are required to complete this step.

Table 3. Parameter for `archive:port-to-apigw`

Parameter	Description
<code><path></code>	<p>Specify the unpacked <code>cloud</code> directory as input:</p> <pre>./apicm archive:port-to-apigw /cloud</pre> <p>By specifying <code>/cloud</code>, <code>apicm</code> ports all artifacts within the directory, including APIs, Products, OAuth providers, custom policies, subscriptions and so on. Running a subsequent <code>archive:push</code> pushes all the artifacts to API Gateway.</p> <p>Running <code>archive:port-to-apigw</code> on any subdirectory within the unpacked <code>/cloud</code> directory could leave the directory in an inconsistent state, and cause the subsequent <code>archive:push</code> of artifacts to API Gateway to fail. Custom policies and OAuth providers are not migrated if a directory other than <code>/cloud</code> is specified.</p> <p>Note: However, if you decide to rewrite one API at a time, to migrate them to API Gateway, you can use:</p> <pre>./apicm archive:port-to-apigw <path></pre> <p>where <code><path></code> is a single API file, or a directory that contains only APIs or Products.</p> <p>In this case, you cannot use <code>archive:push</code> to push artifacts to API Gateway. All the ported APIs and Products must be pushed (or imported) into API Manager by either the toolkit or UI.</p>

Usage notes:

- The conversion produces a draft output YAML file. Review the entire draft output YAML file for conformance, performance, and errors. Review the log files for messages about any manual actions that might be needed. For more information, see [Reviewing APIs converted to DataPower API Gateway](#).
- By default, all changes that the migration utility makes when rewriting APIs are logged as comments within the API YAML file. You can use the `enable-api-logging` parameter to disable logging of these changes.

```
./apicm archive:port-to-apigw <folder> --enable-api-logging=false
```

- You can also specify migration configuration parameters using either a YAML configuration file or from the command line, but not both. For information about configuration options and DataPower requirements for certain configuration parameters, see [Configuring migration options for DataPower API Gateway](#).
 - Multiple runs of `./apicm archive:port-to-apigw` on the same `/cloud` folder will skip artifacts that have already been converted to DataPower® API Gateway.
 - If your v10 deployment uses DataPower MultiProtocol Gateway (v5-compatible) you do not need to convert the API definitions.
10. Use `apicm` commands to push the Cloud Manager admin organization data onto your v10 Management Server.
- Log in to the Cloud Manager on the destination v10 Management Server as an Owner of the admin organization:

Important: You must be logged in as an Owner of the admin organization to be able to migrate Cloud Manager admin organization data.

 - Syntax:

```
./apicm login --server <cloud_manager_endpoint> --realm admin/identity_provider --username admin --password admin_password
```

- Example on API Connect v10 standalone

```
./apicm login --server cloud-admin-ui.mgmt.dev.apic10-stack.example.test --realm admin/default-idp-1 --username admin --password 'myadminpassword'
```

- Example on Cloud Pak for Integration

```
./apicm login --server cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod --realm admin/default-idp-1 --username admin --password 'XXXXXXXXXXXX'
```

The values to supply for the login parameters depend on whether your management server is deployed on API Connect v10 standalone or as part of Cloud Pak for Integration. See [Table 4](#) to review how to obtain each parameter.

Note: If you want to login using an OIDC user registry, see [Logging in to a management server with an OIDC registry](#).

Table 4. `apicm login` command parameters

Parameters	Description						
<code>--server <cloud_manager_endpoint></code>	<p>The endpoint on the management server for communication with the Cloud Manager user interface.</p> <p>v10 standalone</p> <p>Example:</p> <pre>cloud-admin-ui.mgmt.dev.apic10-stack.example.test</pre> <p>The endpoint was specified when the management subsystem was installed. View the <code>apiconnect_api</code> definition in the <code>apiconnect-up.yml</code> file in the API Connect v10 installation project directory that you created for initial deployment of v10.</p> <p>Examples:</p> <ul style="list-style-type: none"> VMware: see the <code>cloud-admin-ui</code> endpoint in Deployment overview for endpoints and certificates. Kubernetes: see the <code>admin</code> endpoint in Deployment overview for endpoints and certificates. <p>Cloud Pak for Integration</p> <p>Syntax:</p> <pre><cpd-APIC_namespace.your-company.domain.com>/integration/apis/<APIC_namespace>/<APIC_instance></pre> <p>Example:</p> <pre>cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod</pre> <p>where <code>e2e</code> is <code><APIC_namespace></code> and <code>prod</code> is <code><APIC_instance></code>.</p> <p>Note: How to obtain the server parameter on Cloud Pak for Integration</p> <p>Use the Platform Navigator UI to determine the parameter. Note that you cannot get the endpoint by using <code>oc get routes</code> because Zen modifies the route internally.</p> <ol style="list-style-type: none"> Locate the Platform UI panel with entries for the deployment. Identify the row that has a Type column entry with the description <code>API management administration</code> To display the endpoint, hover over the entry in the Name column for that row. For example, hover over the <code>prod</code> entry, as shown in bold font, and the <code>URL</code> displays at the bottom of the panel. <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>prod</td> <td>API management administration</td> </tr> <tr> <td>prod</td> <td>API management</td> </tr> </tbody> </table> <pre>cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod/admin</pre> <ol style="list-style-type: none"> Remove the trailing <code>/admin</code> from the endpoint, to obtain the parameter for the login command: <pre>cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod</pre>	Name	Type	prod	API management administration	prod	API management
Name	Type						
prod	API management administration						
prod	API management						

Parameters	Description
<code>--realm</code> <code>admin/default-idp-1</code>	<p>The <code>apicm login</code> example shows the expected <code>--realm</code> parameter values for a default API Connection administration organization. The default user is <code>admin</code> and the identity provider is <code>default-idp-1</code>.</p> <p>You can determine which identity provider to use in the <code>--realm</code> parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):</p> <pre>apicm identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm</pre> <p>For example:</p> <pre>apicm identity-providers:list --scope admin --server myserver.com --fields title,realm total_results: 2 results: - title: Cloud Manager User Registry realm: admin/default-idp-1 - title: Corporate LDAP user registry realm: admin/corporate-ldap</pre> <p>The <code>title</code> value should enable you to determine which identity provider to use; you can then copy the corresponding <code>--realm</code> parameter directly from the displayed <code>realm</code> value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is <code>default-idp-1</code>.</p>
<code>--username</code> <code>admin</code>	Administrative user with permissions to modify API Connect admin-org. The default user is <code>admin</code> .
<code>--password</code> <code>myadminpassword</code>	Administrative user password. Cloud Pak for Integration: When you installed with the top-level CR for CP4I, the password was auto-generated. To obtain the password, see step 3 in Installing API Connect .

b. Run `/apicm archive:push` to migrate data for the Cloud Manager administrative org (admin-org).

- Syntax

```
./apicm archive:push <cloud_manager_endpoint> <path_to_admin_org> [optional flags]
```

- Example for v10:

```
./apicm archive:push cloud-admin-ui.mgmt.dev.apic10-stack.example.test cloud/admin-org
```

- Example for Cloud Pak for Integration:

```
./apicm archive:push cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod cloud/admin-org
```

See [Table 5](#) to review how to obtain each parameter.

Tip: First run the command with the optional flag `--dry-run` to test for errors or warnings. Review the command parameters, examples and guidance that follow before you run it.

Table 5. `apicm archive:push` command parameters for admin org

Parameters	Description
<code><cloud_manager_endpoint></code>	<p>The endpoint on the management server for communication with the Cloud Manager user interface.</p> <p>v10 standalone Example: <code>cloud-admin-ui.mgmt.dev.apic10-stack.example.test</code></p> <p>The endpoint was specified when the management subsystem was installed. View the <code>apiconnect_api</code> definition in the <code>apiconnect-up.yml</code> file in the API Connect v10 installation project directory that you created for initial deployment of v10. Examples:</p> <ul style="list-style-type: none"> • VMware: see the <code>cloud-admin-ui</code> endpoint in Deployment overview for endpoints and certificates. • Kubernetes: see the <code>admin</code> endpoint in Deployment overview for endpoints and certificates. <p>Cloud Pak for Integration Syntax: <code><cpd-APIC_namespace.your-company.domain.com>/integration/apis/<APIC_namespace>/<APIC_instance></code></p> <p>Example: <code>cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod</code></p> <p>where <code>e2e</code> is <code><APIC_namespace></code> and <code>prod</code> is <code><APIC_instance></code>.</p> <p>See How to obtain the server parameter for Cloud Pak for Integration.</p>
<code><path_to_admin_in_org></code>	Must be <code>cloud/admin-org</code> .
<code>--dry-run</code>	<p>Optional flag. You can test the Admin org migration output first view by using the <code>--dry-run</code> option.</p> <pre>./apicm archive:push <cloud_manager_endpoint> <path_to_admin_org> --dry-run</pre> <p>This option creates a project directory similar to <code>cloud</code>, titled <code>.workdir-<apim_admin_server></code> where you can view how the data will be loaded into your new system. The option also displays the objects that will fail due to existing already or missing dependencies. Use this option to view the resolved URLs for where the data will be loaded, without creating resources on the new system. See apicm command line help.</p>
<code>--apigw-only</code>	<p>Optional flag. Use when migrating to a DataPower API Gateway. For example:</p> <pre>./apicm archive:push <cloud_manager_endpoint> <path_to_admin_org> --apigw-only</pre>

Important: Usage notes:

- Run `./apicm archive:push` in the same directory where you ran `./apicm: archive:unpack`.
- If you run migration and observe error messages in your log output, you can take troubleshooting steps, and run the migration again. See [Migration troubleshooting](#).
- The `apicm` login obtains a token that expires after eight hours. If the token expires while `apicm archive:push` is running, `apicm` generates errors due to lack of the necessary access permissions. In this case, log in again and re-run `apicm archive:push`. The token is refreshed every time you log in. See also [Token expiration in Migration troubleshooting](#).
- An alternative method of viewing endpoints is to run `kubectl get ingress` and get the values from the output. For example, see the values in **bold** in the following sample output:

```

r7c221d8a33-apic-portal-director      api.portal.dev.apic10-stack.example.test      80, 443 21h
r7c221d8a33-apic-portal-web          portal.dev.apic10-stack.example.test          80, 443 21h
re314d70920-apiconnect-api           platform.mgmt.dev.apic10-stack.example.test   80, 443 22h
re314d70920-apiconnect-apim-ui       apim.mgmt.dev.apic10-stack.example.test       80, 443 22h
re314d70920-apiconnect-cm-ui         cloud.mgmt.dev.apic10-stack.example.test      80, 443 22h
re314d70920-apiconnect-consumer-api  consumer.mgmt.dev.apic10-stack.example.test   80, 443 22h
rfe867a858b-dynamic-gateway-service  service.gw.dev.apic10-stack.example.test      80, 443 21h
rfe867a858b-dynamic-gateway-service-gw api.gw.dev.apic10-stack.example.test          80, 443 21h
rfff5043330-analytics-client         ac.dev.apic10-stack.example.test             80, 443 21h
rfff5043330-analytics-ingestion      ai.dev.apic10-stack.example.test             80, 443 21h

```

11. Migrate the API Manager provider organizations.

- If you have not already done so, create the provider organizations on your API Connect v10 deployment.
- Use `apicm` to log in to the API Manager user interface as an Owner or Administrator of the provider organization that you want to migrate. If you want to login using an OIDC user registry, see [Logging in to a management server with an OIDC registry](#).

Important: You cannot migrate a provider organization while you are logged in with the Cloud Manager admin organization. You must be logged in as an Owner or Administrator of the provider organization to be able to migrate that organization's data.

- Syntax:

```
./apicm login --server <api-manager-ui-endpoint> --realm provider/identity_provider --username org_username --password org_password
```

- Example for API Connect v10:

```
./apicm login --server api-manager-ui.dev.apic10-stack.example.test --realm provider/default-idp-2 --username org_username@example.com --password myorgpassword
```

- Example for Cloud Pak for Integration:

```
./apicm login --server cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod --realm provider/default-idp-2 --username org_username@example.com --password myorgpassword
```

Table 6. `apicm login` parameters for logging in to provider orgs on API Connect v10 standalone

Parameters	Description
<code>--server <api-manager-ui-endpoint></code>	<p>API Manager URL endpoint on the management server for communication with the API Manager user interface.</p> <p>API Connect v10 standalone</p> <p>For example:</p> <pre>api-manager-ui.dev.apic10-stack.example.test</pre> <p>The endpoint was specified when the management subsystem was installed. To verify your endpoint setting you can:</p> <ul style="list-style-type: none"> • View the <code>apiconnect-apim-ui</code> definition in the <code>apiconnect-up.yml</code> file in the API Connect v10 installation project directory. • In the Cloud Manager UI, select Settings > Endpoints and view API Manager URL. <p>Cloud Pak for Integration</p> <p>Syntax (line-break is for table formatting only; your value will not include a line-break):</p> <pre><cpd-APIC_namespace.your-company.domain.com>/integration/apis/<APIC_namespace>/<APIC_instance></pre> <p>Example:</p> <pre>cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod</pre> <p>where <code>e2e</code> is <code><APIC_namespace></code> and <code>prod</code> is <code><APIC_instance></code>.</p> <p>See How to obtain the server parameter for Cloud Pak for Integration.</p>

Parameters	Description
<pre>--realm provider/identity_provider</pre>	<p>The <code>apicm login</code> example shows the expected <code>realm</code> parameter values for a default API Connection provider organization. The default user is <code>provider</code> and the identity provider is <code>default-idp-2</code>.</p> <p>You can determine which identity provider to use in the <code>--realm</code> parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):</p> <pre>apicm identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm</pre> <p>For example,</p> <pre>apicm identity-providers:list --scope provider --server myserver.com --fields title,realm total_results: 2 results: - title: Cloud Manager User Registry realm: provider/default-idp-2 - title: Corporate LDAP user registry realm: provider/corporate-ldap</pre> <p>The <code>title</code> value should enable you to determine which identity provider to use; you can then copy the corresponding <code>--realm</code> parameter directly from the displayed <code>realm</code> value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is <code>default-idp-2</code>.</p>
<pre>--username <org_username></pre>	A user with permissions to modify the provider organization. Typically this is the owner or administrator of the organization.
<pre>--password <org_password></pre>	Organization owner or administrator password.

c. Run `/apicm archive:push` to migrate data for the Provider organization.

- Syntax:

```
./apicm archive:push <api-manager-ui-endpoint> cloud/provider-orgs/<pOrg_name> [optional flags]
```

- Example on v10:

```
./apicm archive:push api-manager-ui.mgmt.dev.apic10-stack.example.test cloud/provider-orgs/production_Org
```

- Example on Cloud Pak for Integration

```
./apicm archive:push cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod cloud/provider-orgs/production_Org
```

See [Table 7](#) to review how to obtain each parameter.

Tip: First run the command with the optional flag `--dry-run` to test for errors or warnings.

Table 7. `apicm archive:push` command parameters for Provider orgs

Parameters	Description
<pre><api-manager-ui-endpoint></pre>	<p>API Manager URL endpoint on the management server for communication with the API Manager user interface.</p> <p>API Connect v10 standalone</p> <p>For example:</p> <pre>api-manager-ui.dev.apic10-stack.example.test</pre> <p>The endpoint was specified when the management subsystem was installed. To verify your endpoint setting you can:</p> <ul style="list-style-type: none"> • View the <code>apiconnect_api</code> definition in the <code>apiconnect-up.yml</code> file in the API Connect v10 installation project directory. • In the Cloud Manager UI, select Settings > Endpoints and view API Manager URL. <p>Cloud Pak for Integration</p> <p>Syntax (line-break is for table formatting only, your value will not include a line-break).</p> <pre><cpd-APIC_namespace.your-company.domain.com>/integration/apis /APIC_namespace/<APIC_instance></pre> <p>Example:</p> <pre>cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod</pre> <p>where <code>e2e</code> is <code><APIC_namespace></code> and <code>prod</code> is <code><APIC_instance></code>.</p> <p>See How to obtain the server parameter for Cloud Pak for Integration.</p>
<pre>cloud/provider-orgs/<pOrg_name></pre>	<p>The location of your Provider org within the <code>/cloud</code> file hierarchy you created when you ran <code>apicm archive:unpack</code>.</p> <pre>cloud/provider-orgs/production_Org</pre>

Parameters	Description
<code>--catalogs=<name></code>	<p>Optionally, you can push data for only specified Catalogs if you don't want to push the entire Provider organization. Use this flag to specify the name of the Catalog you want to push:</p> <pre>./apicm archive:push <api-manager-ui-endpoint> cloud/provider-orgs/<pOrg_name> --catalogs=<catalog_name></pre> <p>Note: Type the command on a single line. When a catalog is mapped, specify the mapped new catalog name.</p> <p>To specify multiple catalogs, use a comma-separated list:</p> <pre>./apicm archive:push <api-manager-ui-endpoint> cloud/provider-orgs/<pOrg_name> --catalogs=<catalog_name1,catalog_name2,catalog_name3></pre> <p>Note: Type the command on a single line.</p>
<code>--dry-run</code>	<p>You can test the Provider org migration output first view by using the <code>--dry-run</code> option. This will create a project directory similar to <code>cloud</code>, titled <code>.workdir-<apim_admin_server></code> where you can view how the data will be loaded into your new system. The option also displays the objects that will fail due to existing already or missing dependencies. Use this option to view the resolved URLs for where the data will be loaded, without creating resources on the new system. See apicm command line help.</p> <pre>./apicm archive:push <api-manager-ui-endpoint> cloud/provider-orgs/<pOrg_name> --dry-run</pre> <p>Note: Type the command on a single line.</p>
<code>--no-drafts</code>	<p>Do not push any Draft APIs or Products. Draft objects are objects that have not been published.</p> <pre>./apicm archive:push <cloud_manager_endpoint> <path_to_provider_org> --no-drafts</pre> <p>Note: Type the command on a single line. This option can be used with or without <code>--catalogs=a,b,c</code>.</p>
<code>--overwrite</code>	<p>When this option is used, draft products, draft APIs, published products and their APIs, and OAuth Providers that are already on the v10 system will be overwritten. When this option is not used these artifacts will be skipped.</p>

Important: Usage notes

- Run `./apicm archive:push` in the same directory where you ran `./apicm:archive:unpack`.
- If you run migration and observe error messages in your log output, you can take troubleshooting steps, and run the migration again. See [Migration troubleshooting](#).
- The `apicm` login obtains a token that expires after eight hours. If the token expires while `apicm archive:push` is running, `apicm` generates errors due to lack of the necessary access permissions. In this case, log in again and re-run `apicm archive:push`. The token is refreshed every time you log in. See also [Token expiration in Migration troubleshooting](#)
- An alternative method of viewing endpoints is to run `kubectl get ingress` and get the values from the output. See [kubectl example](#).

12. When migration finishes, complete the following steps:

- a. If a user in a Local User Registry is owner of a Consumer Organization, the user must reset their password.
- b. If applicable, reapply any customized configurations from your v5 deployment. See [Customizations to recreate](#).
- c. Optional: If necessary, switch the provider organization ownership to the identity that owned it in v5. This step might be necessary if you created and migrated the provider organization using a different identity (such as administrative account) than the user identity of the v5 provider org. Note that the UI does not provide this function, but you can use the API `/orgs/{org}/transfer-owner` to switch ownership.
- d. If Stripe billing integrations were included in your migration, you must now configure the Stripe integrations in your Developer Portal sites; see [Configuring Stripe in the Developer Portal](#) for detailed instructions. For more information about the billing feature in v10, see [Monetizing your Products](#).

Migration steps without PDUR

Complete the following steps to migrate from v5 to v10 when your v5 catalogs do not use Portal Delegated User Registries (PDUR).

Before you begin

Important: Do not use these instructions if your v5 catalogs use Portal Delegated User Registry. Instead, see [Migration steps with PDUR](#).

- Ensure you planned your migration, obtained the migration utility, and installed a v10 deployment, as described in [Migration steps](#)
- Review [Migration considerations](#).
- Ensure you have system administrator privileges.
- To use `apicm` to convert the API definitions, you must have Node version 18.17.1 (or greater) installed.

About this task

Use the `apicm` utility to migrate your API Connect v5 data to a new API Connect v10 installation.

Tip:

To achieve a smooth and successful migration, we recommend these best practices:

- Before you push your v5 data to v10, your v10 system must be up and running, with gateway services, provider organizations, user registries, etc, ready to receive the data.
- Plan to do several test iterations of migration. You can use the `--dry-run` option to do iterations of the migration push, to test for any errors or warnings to address before the actual data push.
- Use the `apicm <command> --help` command to review the command options and their purpose. To see the list of commands you can run: `apicm help`.

- Complete all the steps in this topic in the order listed, unless the step is marked optional and does not apply to your scenario. You can push provider orgs as many times as needed.

Procedure

1. Log in to the command line interface of the v5 appliance. Use the following command to create a tar.gz export of the v5 configuration database and place it on an **sftp** server:

```
config dbextract sftp <host_name> user <user_name> file <path/name>
```

- Specify a name of your choice for the compressed file to contain the dbextract. For example, **v5mgmt_data.tar.gz**.
- If you are not familiar with the v5 appliance CLI, see [The Command Line Interface](#). For information on **config dbextract**, see [Configuration commands](#).

2. Optional: Setup a temporary SMTP server.

During migration, if subscription approval is needed to enable an application to subscribe to a plan, notifications are sent to all users, within the organization, who have subscription approval permission. In some cases, this results in a large number of emails. To avoid the sending of these emails, set up a temporary email server prior to running the migration, and use it to collect all notifications rather than sending them to users' email accounts.

If subscription approvals are not needed during your migration, you can skip this step.

3. Optional: If necessary, set up a temporary LDAP server. This can be needed if you are temporarily behind a firewall.

- a. Install a server:

```
git clone https://github.com/veo-labs/ldap-server-mock
npm install
```

- b. Creating the configuration files:

- **ldap-server-mock-conf.json**

```
{
  "port": 3004,
  "userLoginAttribute": "cn",
  "searchBase": "dc=test",
  "searchFilter": "(&(objectclass=person)(cn={username}))"
}
```

- **users.json**

```
[
  {
    "dn": "cn=user,dc=test",
    "cn": "user-login"
  }
]
```

- c. Run the LDAP server:

```
node server.js --conf=./ldap-server-mock-conf.json --database=./users.json
```

- d. Modifying a current LDAP provider.

The following file is a dummy user registry file that maps to a read-only test server. The fields in bold font (**admin_dn** and **endpoint**) have been changed to fit the dummy server configuration. Change these on all user registries that need to be fixed.

```
api_version: 2.0.0
configuration:
  admin_dn: cn=user,dc=test
  admin_password: xxxx
  authenticated_bind: "true"
  authentication_method: search_dn
  bind_prefix: ""
  bind_suffix: ""
  protocol_version: "3"
  search_dn_base: ""
  search_dn_filter_prefix: (examplelegacyuid=
  search_dn_filter_suffix: )
  search_dn_scope: sub
endpoint:
  endpoint: ldap://172.20.20.11:3004
  tls_client_profile_url: idurl://...\\admin-org\\tls-client-profiles\\tls-profile-for-ldap-1.0.0.yml
integration_url: file://...\\integrations\\user_registry\\ldap.yml
metadata:
  v5_id: xxxxxx40cf2fc09a9ba0d6c
  v5_scope: ["api"]
  v5_type: ldap
name: xxxxxxxxxxxxxxxx
title: xxxxxxxxxxxxxxxxxxxxxxxx
type: user_registry
```

4. Unpack the V5 configuration data onto the local filesystem.

Important: Migration on Windows 10

Windows has path length limits that can interfere with large extracts. It is recommended that you keep the initial path for migration on Windows as short as possible. The AMU must be run in a command window with administrative privileges on Windows. Running from a Linux VM on your Windows system may be a better option for doing migration.

- a. Copy the v5 extract tar file from the sftp server to the location where you have v10 migration utility installed.
- b. Run **apicm archive:unpack** command from the directory where you downloaded the v5 extract file.

```
./apicm archive:unpack <v5_extract_file>.tgz
```

When you invoke **apicm** for the first time, you are prompted to accept the license. Accept the license to continue.

You can optionally limit the unpack to only specific provider orgs. See [Table 1](#).

Table 1. `apicm archive:unpack` command options

Parameters	Description
<code><v5_extract_file></code>	Can be compressed file (.tgz, tar.gz) or uncompressed file (.tar). Use any file name of your choosing. Example: <code>./apicm archive:unpack v5_DBextract_data.tgz</code>
<code>--provider-orgs</code>	One or more provider orgs to unpack. This flag is optional. Use it when you want to unpack only specific provider orgs rather than the entire archive. Example: <code>./apicm archive:unpack v5_DBextract_data.tgz --provider-orgs=<pOrg1_name></code> Use a comma-separated list to specify multiple provider orgs: <code>./apicm archive:unpack v5_archive_data.tgz --provider-orgs=<pOrg1_name>,<pOrg2_name>,<pOrg3_name></code>
<code>--use-old-sluggify</code>	Unpack the V5 configuration data onto the local filesystem using the version 10.0.5.1 conversion rules for sluggifying element names and version numbers. For more information about the conversion rules, see Parity between v5 and v10 .

Note:

- **Important:** Any Warnings or Errors that occur during `archive:unpack` should be investigated for remediation. See [Migration troubleshooting](#)
- You can avoid the license prompt by adding `--accept-license` to the command line.
- The unpack command ensures that catalog names and API properties contain only valid characters. Any invalid characters are mapped to '-'. See [Parity between v5 and v10](#).
- The unpack command issues warning messages for APIs and Products that fail validation against the OpenAPI specification. See the limitations section in [Migration considerations](#).
- For troubleshooting tips, see [Migration troubleshooting](#)
- The unpack command might return error messages related to PDUR, such as:

```
ERRO[3414] PDUR backup data for IDP PortalDelegatedIdentityProvider-123b81fbe4b06de411580e2e
(123b81fbe4b06de411580e2f) not provided: not found
ERRO[3414] PDUR backup data for IDP PortalDelegatedIdentityProvider-123b820de4b06de411580e32
(123b820ee4b06de411580e33) not provided: not found
```

If you are not migrating PDUR backup data, and your command did not include a `<pdur_user_export>.tgz`, the error indicates that you do in fact have a PDUR registry, and that you should have included a PDUR backup. Follow the migration instructions in [Migration steps with PDUR](#).

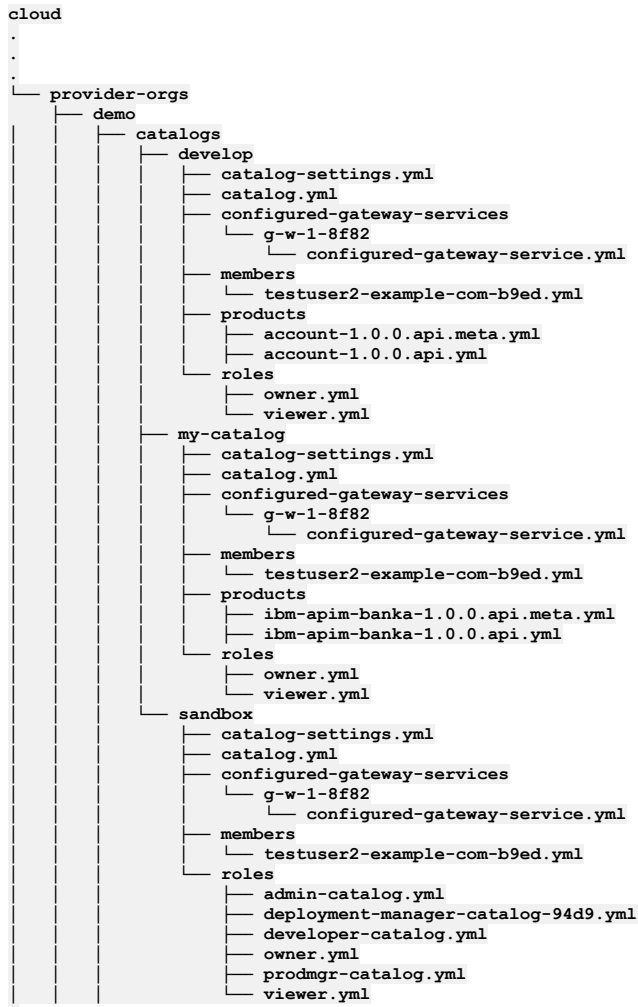
When the command completes, the data is unpacked to the `/cloud` directory. The `/cloud` directory is located in the same directory as `apicm`, and populated with yaml files representing the configuration objects in the export. The unpack processes produce output in the format required for v10 configuration.

Note: The unpack directory must be named `/cloud`. Do not rename the directory.

Example output in `/cloud` for `admin-org`:

```
cloud
├── admin-org
│   ├── availability-zones
│   │   ├── availability-zone-default
│   │   │   ├── availability-zone.yml
│   │   │   └── gateway-services
│   │   │       └── g-w-1-8f82
│   │   │           └── gateway-service.yml
│   ├── keystore
│   │   └── default-ssl-profile.yml
│   ├── provider-org.yml
│   ├── tls-client-profiles
│   │   ├── default-ssl-profile-1.0.0.yml
│   │   ├── new-tls-profile-1-1.0.0.yml
│   │   └── testclientprofile-1.0.0.yml
│   ├── tls-server-profiles
│   │   └── default-ssl-profile-1.0.0.yml
│   ├── truststore
│   │   └── default-ssl-profile.yml
│   └── user-registries
│       ├── api-manager-lur
│       │   ├── user-registry.yml
│       │   └── users
│       │       ├── adminuser1-catalog1-com-3ac2.yml
│       │       └── adminuser2-catalog1-com-39c1.yml
│       ├── cloud-manager-lur
│       │   ├── user-registry.yml
│       │   └── users
│       │       └── admin.yml
│       ├── ldap1
│       │   ├── user-registry.yml
│       │   └── users
│       │       └── user1-gmail-com-47e5.yml
│       ├── oauth-ldap
│       │   └── user-registry.yml
│       ├── user-ldap
│       │   ├── user-registry.yml
│       │   └── users
│       │       └── testuser1.yml
│       └── vrp-authurl
│           └── user-registry.yml
├── .
└── .
```

Example output in `/cloud` for `provider-orgs`:



5. Remove unneeded configuration objects.

Inspect the set of configuration objects and their attributes, and remove the ones that are not needed. In many cases, some of the configuration objects present in the v5 system are not needed in the new v10 environment. For example, there is no value in preserving inactive users, obsolete apps, obsolete consumer organizations, and so on.

When deleting objects, make sure that you do not delete objects that have other dependent objects, such as products with subscriptions on them or users that are active. If you delete a consumer organization make sure you also delete all apps and subscriptions associated with that consumer org. Be aware of any groups that include these consumer orgs or visibility settings that include the consumer orgs.

6. Map v5 object names to v10 object names.

Most deployments adjust the names or quantities of gateway services, provider organizations, and user registries when setting up a new environment for v10. Be sure to specify any changed v10 destinations for the v5 data in a migration mapping file so that the migration utility can automatically move the data.

Complete the following, as needed. You can skip the steps for any objects that are unchanged from v5 to v10.

- a. [What is a mapping file?](#)
- b. [Mapping a gateway service](#)
- c. [Mapping a provider organization](#)

7. Optional: If your v10 deployment uses a **DataPower API Gateway**, convert the v5 API definitions to v10 DataPower® API Gateway.

To convert API definitions using the default migration configuration, use the following syntax.

```
./apicm archive:port-to-apigw <path>
```

Important: Ensure that the DataPower Gateway version that you are using with your API Connect v10 deployment is compatible with the version of the migration utility that you are using. For details, see the version compatibility information in [Configuring and managing your server environment](#).

Important: If you are running the migration utility in Windows, Administrator privileges are required to complete this step.

Table 2. Parameter for archive:port-to-apigw

Parameter	Description
-----------	-------------

Parameter	Description
<code><path></code>	<p>Specify the unpacked <code>cloud</code> directory as input:</p> <pre>./apicm archive:port-to-apigw /cloud</pre> <p>By specifying <code>/cloud</code>, <code>apicm</code> ports all artifacts within the directory, including APIs, Products, OAuth providers, custom policies, subscriptions and so on. Running a subsequent <code>archive:push</code> pushes all the artifacts to API Gateway.</p> <p>Running <code>archive:port-to-apigw</code> on any subdirectory within the unpacked <code>/cloud</code> directory could leave the directory in an inconsistent state, and cause the subsequent <code>archive:push</code> of artifacts to API Gateway to fail. Custom policies and OAuth providers are not migrated if a directory other than <code>/cloud</code> is specified.</p> <p>Note: However, if you decide to rewrite one API at a time, to migrate them to API Gateway, you can use:</p> <pre>./apicm archive:port-to-apigw <path></pre> <p>where <code><path></code> is a single API file, or a directory that contains only APIs or Products.</p> <p>In this case, you cannot use <code>archive:push</code> to push artifacts to API Gateway. All the ported APIs and Products must be pushed (or imported) into API Manager by either the toolkit or UI.</p>

Usage notes:

- The conversion produces a draft output YAML file. Review the entire draft output YAML file for conformance, performance, and errors. Review the log files for messages about any manual actions that might be needed. For more information, see [Reviewing APIs converted to DataPower API Gateway](#).
 - By default, all changes that the migration utility makes when rewriting APIs are logged as comments within the API YAML file. You can use the `enable-api-logging` parameter to disable logging of these changes.
- ```
./apicm archive:port-to-apigw <folder> --enable-api-logging=false
```
- You can also specify migration configuration parameters using either a YAML configuration file or from the command line, but not both. For information about configuration options and DataPower requirements for certain configuration parameters, see [Configuring migration options for DataPower API Gateway](#).
  - Multiple runs of `./apicm archive:port-to-apigw` on the same `/cloud` folder will skip artifacts that have already been converted to DataPower® API Gateway.
  - If your v10 deployment uses DataPower MultiProtocol Gateway (v5-compatible) you do not need to convert the API definitions.
8. Use `apicm` commands to push the Cloud Manager admin organization data onto your v10 Management Server.
- a. Log in to the Cloud Manager on the destination v10 Management Server as an Owner of the admin organization:
- Important: You must be logged in as an Owner of the admin organization to be able to migrate Cloud Manager admin organization data.

- Syntax:

```
./apicm login --server <cloud_manager_endpoint> --realm admin/identity_provider --username admin --password admin_password
```

- Example on API Connect v10 standalone

```
./apicm login --server cloud-admin-ui.mgmt.dev.apic10-stack.example.test --realm admin/default-idp-1 --username admin --password 'myadminpassword'
```

- Example on Cloud Pak for Integration

```
./apicm login --server cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod --realm admin/default-idp-1 --username admin --password 'XXXXXXXXXXXX'
```

The values to supply for the login parameters depend on whether your management server is deployed on API Connect v10 standalone or as part of Cloud Pak for Integration. See [Table 3](#) to review how to obtain each parameter.

Note: If you want to login using an OIDC user registry, see [Logging in to a management server with an OIDC registry](#).

Table 3. `apicm login` command parameters

| Parameters | Description |
|------------|-------------|
|------------|-------------|

| Parameters                                         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |      |      |             |                               |      |                |
|----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|-------------|-------------------------------|------|----------------|
| <pre>--server &lt;cloud_manager_endpoint&gt;</pre> | <p>The endpoint on the management server for communication with the Cloud Manager user interface.</p> <p>v10 standalone<br/>Example:</p> <pre>cloud-admin-ui.mgmt.dev.apic10-stack.example.test</pre> <p>The endpoint was specified when the management subsystem was installed. View the <code>apiconnect_api</code> definition in the <code>apiconnect-up.yml</code> file in the API Connect v10 installation project directory that you created for initial deployment of v10.<br/>Examples:</p> <ul style="list-style-type: none"> <li>VMware: see the <code>cloud-admin-ui</code> endpoint in <a href="#">Deployment overview for endpoints and certificates</a>.</li> <li>Kubernetes: see the <code>admin</code> endpoint in <a href="#">Deployment overview for endpoints and certificates</a>.</li> </ul> <p>Cloud Pak for Integration<br/>Syntax:</p> <pre>&lt;cpd-APIC_namespace.your-company.domain.com&gt;/integration/apis/&lt;APIC_namespace&gt;/&lt;APIC_instance&gt;</pre> <p>Example:</p> <pre>cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod</pre> <p>where <code>e2e</code> is <code>&lt;APIC_namespace&gt;</code> and <code>prod</code> is <code>&lt;APIC_instance&gt;</code>.</p> <p>Note: How to obtain the server parameter on Cloud Pak for Integration<br/>Use the Platform Navigator UI to determine the parameter. Note that you cannot get the endpoint by using <code>oc get routes</code> because Zen modifies the route internally.</p> <ol style="list-style-type: none"> <li>Locate the Platform UI panel with entries for the deployment. Identify the row that has a Type column entry with the description <code>API management administration</code></li> <li>To display the endpoint, hover over the entry in the Name column for that row. For example, hover over the <code>prod</code> entry, as shown in bold font, and the <code>URL</code> displays at the bottom of the panel.</li> </ol> <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td><b>prod</b></td> <td>API management administration</td> </tr> <tr> <td>prod</td> <td>API management</td> </tr> </tbody> </table> <pre>cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod/admin</pre> <ol style="list-style-type: none"> <li>Remove the trailing <code>/admin</code> from the endpoint, to obtain the parameter for the login command:</li> </ol> <pre>cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod</pre> | Name | Type | <b>prod</b> | API management administration | prod | API management |
| Name                                               | Type                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |      |      |             |                               |      |                |
| <b>prod</b>                                        | API management administration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |      |      |             |                               |      |                |
| prod                                               | API management                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |      |      |             |                               |      |                |
| <pre>--realm admin/default-idp-1</pre>             | <p>The <code>apicm login</code> example shows the expected <code>--realm</code> parameter values for a default API Connection administration organization. The default user is <code>admin</code> and the identity provider is <code>default-idp-1</code>.<br/>You can determine which identity provider to use in the <code>--realm</code> parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):</p> <pre>apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm</pre> <p>For example:</p> <pre>apic identity-providers:list --scope admin --server myserver.com --fields title,realm total_results: 2 results: - title: Cloud Manager User Registry   realm: admin/default-idp-1 - title: Corporate LDAP user registry   realm: admin/corporate-ldap</pre> <p>The <code>title</code> value should enable you to determine which identity provider to use; you can then copy the corresponding <code>--realm</code> parameter directly from the displayed <code>realm</code> value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is <code>default-idp-1</code>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |      |      |             |                               |      |                |
| <pre>--username admin</pre>                        | <p>Administrative user with permissions to modify API Connect admin-org. The default user is <code>admin</code>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |      |      |             |                               |      |                |
| <pre>--password myadminpassword</pre>              | <p>Administrative user password.<br/><b>Cloud Pak for Integration:</b> When you installed with the top-level CR for CP4I, the password was auto-generated. To obtain the password, see step 3 in <a href="#">Installing API Connect</a>:</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |      |      |             |                               |      |                |

b. Run `/apicm archive:push` to migrate data for the Cloud Manager administrative org (admin-org).

- Syntax

```
./apicm archive:push <cloud_manager_endpoint> <path_to_admin_org> [optional flags]
```

- Example for v10:

```
./apicm archive:push cloud-admin-ui.mgmt.dev.apic10-stack.example.test cloud/admin-org
```

- Example for Cloud Pak for Integration:

```
./apicm archive:push cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod cloud/admin-org
```

See [Table 4](#) to review how to obtain each parameter.

Tip: First run the command with the optional flag `--dry-run` to test for errors or warnings. Review the command parameters, examples and guidance that follow before you run it.

Table 4. `apicm`  
`archive:push` command parameters for admin org

| Parameters                                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>&lt;cloud_manager_endpoint&gt;</code> | <p>The endpoint on the management server for communication with the Cloud Manager user interface.</p> <p>v10 standalone<br/>           Example:</p> <pre>cloud-admin-ui.mgmt.dev.apic10-stack.example.test</pre> <p>The endpoint was specified when the management subsystem was installed. View the <code>apiconnect_api</code> definition in the <code>apiconnect-up.yml</code> file in the API Connect v10 installation project directory that you created for initial deployment of v10.<br/>           Examples:</p> <ul style="list-style-type: none"> <li>VMware: see the <code>cloud-admin-ui</code> endpoint in <a href="#">Deployment overview for endpoints and certificates</a>.</li> <li>Kubernetes: see the <code>admin</code> endpoint in <a href="#">Deployment overview for endpoints and certificates</a>.</li> </ul> <p>Cloud Pak for Integration<br/>           Syntax:</p> <pre>&lt;cpd-APIC_namespace.your-company.domain.com&gt;/integration/apis/&lt;APIC_namespace&gt;/&lt;APIC_instance&gt;</pre> <p>Example:</p> <pre>cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod</pre> <p>where <code>e2e</code> is <code>&lt;APIC_namespace&gt;</code> and <code>prod</code> is <code>&lt;APIC_instance&gt;</code>.</p> <p>See <a href="#">How to obtain the server parameter for Cloud Pak for Integration</a>.</p> |
| <code>&lt;path_to_admin_org&gt;</code>      | Must be <code>cloud/admin-org</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>--dry-run</code>                      | <p>Optional flag. You can test the Admin org migration output first view by using the <code>--dry-run</code> option.</p> <pre>./apicm archive:push &lt;cloud_manager_endpoint&gt; &lt;path_to_admin_org&gt; --dry-run</pre> <p>This option creates a project directory similar to <code>cloud</code>, titled <code>.workdir-&lt;apim_admin_server&gt;</code> where you can view how the data will be loaded into your new system. The option also displays the objects that will fail due to existing already or missing dependencies. Use this option to view the resolved URLs for where the data will be loaded, without creating resources on the new system. See <a href="#">apicm command line help</a>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>--apigw-only</code>                   | <p>Optional flag. Use when migrating to a DataPower API Gateway. For example:</p> <pre>./apicm archive:push &lt;cloud_manager_endpoint&gt; &lt;path_to_admin_org&gt; --apigw-only</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

Important: Usage notes:

- Run `./apicm archive:push` in the same directory where you ran `./apicm archive:unpack`.
- If you run migration and observe error messages in your log output, you can take troubleshooting steps, and run the migration again. See [Migration troubleshooting](#).
- The `apicm` login obtains a token that expires after eight hours. If the token expires while `apicm archive:push` is running, `apicm` generates errors due to lack of the necessary access permissions. In this case, log in again and re-run `apicm archive:push`. The token is refreshed every time you log in. See also [Token expiration](#) in [Migration troubleshooting](#).
- An alternative method of viewing endpoints is to run `kubectl get ingress` and get the values from the output. For example, see the values in **bold** in the following sample output:

```
r7c221d8a33-apic-portal-director api.portal.dev.apic10-stack.example.test 80, 443 21h
r7c221d8a33-apic-portal-web portal.dev.apic10-stack.example.test 80, 443 21h
re314d70920-apiconnect-api platform.mgmt.dev.apic10-stack.example.test 80, 443 22h
re314d70920-apiconnect-apim-ui apim.mgmt.dev.apic10-stack.example.test 80, 443 22h
re314d70920-apiconnect-cm-ui cloud.mgmt.dev.apic10-stack.example.test 80, 443 22h
re314d70920-apiconnect-consumer-api consumer.mgmt.dev.apic10-stack.example.test 80, 443 22h
rfe867a858b-dynamic-gateway-service service.gw.dev.apic10-stack.example.test 80, 443 21h
rfe867a858b-dynamic-gateway-service-gw api.gw.dev.apic10-stack.example.test 80, 443 21h
rfff5043330-analytics-client ac.dev.apic10-stack.example.test 80, 443 21h
rfff5043330-analytics-ingestion ai.dev.apic10-stack.example.test 80, 443 21h
```

#### 9. Migrate the API Manager provider organizations.

- If you have not already done so, create the provider organizations on your API Connect v10 deployment.
- Use `apicm` to log in to the API Manager user interface as an Owner or Administrator of the provider organization that you want to migrate. If you want to login using an OIDC user registry, see [Logging in to a management server with an OIDC registry](#).

Important: You cannot migrate a provider organization while you are logged in with the Cloud Manager admin organization. You must be logged in as an Owner or Administrator of the provider organization to be able to migrate that organization's data.

- Syntax:

```
./apicm login --server <api-manager-ui-endpoint> --realm provider/identity_provider --username org_username --password org_password
```

- Example for API Connect v10:

```
./apicm login --server api-manager-ui.dev.apic10-stack.example.test --realm provider/default-idp-2 --username org_username@example.com --password myorgpassword
```

- Example for Cloud Pak for Integration:

```
./apicm login -server cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod --realm provider/default-idp-2 --username org_username@example.com --password myorgpassword
```



Table 5. `apicm login` parameters for logging in to provider orgs on API Connect v10 standalone

| Parameters                                            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>--server &lt;api- manager-ui- endpoint&gt;</pre> | <p>API Manager URL endpoint on the management server for communication with the API Manager user interface.</p> <p>API Connect v10 standalone</p> <p>For example:</p> <pre>api-manager-ui.dev.apic10-stack.example.test</pre> <p>The endpoint was specified when the management subsystem was installed. To verify your endpoint setting you can:</p> <ul style="list-style-type: none"> <li>View the <code>apiconnect-apim-ui</code> definition in the <code>apiconnect-up.yml</code> file in the API Connect v10 installation project directory.</li> <li>In the Cloud Manager UI, select Settings, Endpoints and view <b>API Manager URL</b>.</li> </ul> <p>Cloud Pak for Integration</p> <p>Syntax (line-break is for table formatting only; your value will not include a line-break):</p> <pre>&lt;cpd-APIC_namespace.your-company.domain.com&gt;/integration/apis /&lt;APIC_namespace&gt;/&lt;APIC_instance&gt;</pre> <p>Example:</p> <pre>cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod</pre> <p>where <code>e2e</code> is <code>&lt;APIC_namespace&gt;</code> and <code>prod</code> is <code>&lt;APIC_instance&gt;</code>.</p> <p>See <a href="#">How to obtain the server parameter for Cloud Pak for Integration</a>.</p>                                                                                                                                                                          |
| <pre>--realm provider/ide ntity_provid er</pre>       | <p>The <code>apicm login</code> example shows the expected <code>realm</code> parameter values for a default API Connection provider organization. The default user is <code>provider</code> and the identity provider is <code>default-idp-2</code>.</p> <p>You can determine which identity provider to use in the <code>--realm</code> parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):</p> <pre>apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm</pre> <p>For example,</p> <pre>apic identity-providers:list --scope provider --server myserver.com --fields title,realm total_results: 2 results: - title: Cloud Manager User Registry   realm: provider/default-idp-2 - title: Corporate LDAP user registry   realm: provider/corporate-ldap</pre> <p>The <code>title</code> value should enable you to determine which identity provider to use; you can then copy the corresponding <code>--realm</code> parameter directly from the displayed <code>realm</code> value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is <code>default-idp-2</code>.</p> |
| <pre>--username &lt;org_ usernam e&gt;</pre>          | <p>A user with permissions to modify the provider organization. Typically this is the owner or administrator of the organization.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <pre>--password &lt;org_passwor d&gt;</pre>           | <p>Organization owner or administrator password.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

c. Run `/apicm archive:push` to migrate data for the Provider organization.

- Syntax:

```
./apicm archive:push <api-manager-ui-endpoint> cloud/provider-orgs/<pOrg_name> [optional flags]
```

- Example on v10:

```
./apicm archive:push api-manager-ui.mgmt.dev.apic10-stack.example.test cloud/provider-orgs/production_Org
```

- Example on Cloud Pak for Integration

```
./apicm archive:push cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod cloud/provider-
orgs/production_Org
```

See [Table 6](#) to review how to obtain each parameter.

Tip: First run the command with the optional flag `--dry-run` to test for errors or warnings.

Table 6. `apicm archive:push` command parameters for Provider orgs

| Parameters | Description |
|------------|-------------|
|------------|-------------|

| Parameters                                         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>&lt;api-manager-ui-endpoint&gt;</code>       | <p>API Manager URL endpoint on the management server for communication with the API Manager user interface.</p> <p>API Connect v10 standalone<br/>For example:</p> <pre>api-manager-ui.dev.apic10-stack.example.test</pre> <p>The endpoint was specified when the management subsystem was installed. To verify your endpoint setting you can:</p> <ul style="list-style-type: none"> <li>View the <code>apiconnect_api</code> definition in the <code>apiconnect-up.yml</code> file in the API Connect v10 installation project directory.</li> <li>In the Cloud Manager UI, select Settings_&gt; Endpoints and view <b>API Manager URL</b>.</li> </ul> <p>Cloud Pak for Integration<br/>Syntax (line-break is for table formatting only, your value will not include a line-break).</p> <pre>&lt;cpd-APIC_namespace.your-company.domain.com&gt;/integration/apis/APIC_namespace/&lt;APIC_instance&gt;</pre> <p>Example:</p> <pre>cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod</pre> <p>where <code>e2e</code> is <code>&lt;APIC_namespace&gt;</code> and <code>prod</code> is <code>&lt;APIC_instance&gt;</code>.</p> <p>See <a href="#">How to obtain the server parameter for Cloud Pak for Integration</a>.</p> |
| <code>cloud/provider-orgs/&lt;pOrg_name&gt;</code> | <p>The location of your Provider org within the <code>/cloud</code> file hierarchy you created when you ran <code>apicm archive:unpack</code>.</p> <pre>cloud/provider-orgs/production_Org</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>--catalogs=&lt;name&gt;</code>               | <p>Optionally, you can push data for only specified Catalogs if you don't want to push the entire Provider organization. Use this flag to specify the name of the Catalog you want to push:</p> <pre>./apicm archive:push &lt;api-manager-ui-endpoint&gt; cloud/provider-orgs/&lt;pOrg_name&gt; --catalogs=&lt;catalog_name&gt;</pre> <p>Note: Type the command on a single line.<br/>When a catalog is mapped, specify the mapped new catalog name.</p> <p>To specify multiple catalogs, use a comma-separated list:</p> <pre>./apicm archive:push &lt;api-manager-ui-endpoint&gt; cloud/provider-orgs/&lt;pOrg_name&gt; --catalogs=&lt;catalog_name1,catalog_name2,catalog_name3&gt;</pre> <p>Note: Type the command on a single line.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>--dry-run</code>                             | <p>You can test the Provider org migration output first view by using the <code>--dry-run</code> option. This will create a project directory similar to <code>cloud</code>, titled <code>.workdir-&lt;apim_admin_server&gt;</code> where you can view how the data will be loaded into your new system. The option also displays the objects that will fail due to existing already or missing dependencies. Use this option to view the resolved URLs for where the data will be loaded, without creating resources on the new system. See <a href="#">apicm command line help</a>.</p> <pre>./apicm archive:push &lt;api-manager-ui-endpoint&gt; cloud/provider-orgs/&lt;pOrg_name&gt; --dry-run</pre> <p>Note: Type the command on a single line.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>--no-drafts</code>                           | <p>Do not push any Draft APIs or Products. Draft objects are objects that have not been published.</p> <pre>./apicm archive:push &lt;cloud_manager_endpoint&gt; &lt;path_to_provider_org&gt; --no-drafts</pre> <p>Note: Type the command on a single line.<br/>This option can be used with or without <code>--catalogs=a,b,c</code>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>--overwrite</code>                           | <p>When this option is used, draft products, draft APIs, published products and their APIs, and OAuth Providers that are already on the v10 system will be overwritten. When this option is not used these artifacts will be skipped.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

Important: Usage notes

- Run `./apicm archive:push` in the same directory where you ran `./apicm archive:unpack`.
- If you run migration and observe error messages in your log output, you can take troubleshooting steps, and run the migration again. See [Migration troubleshooting](#).
- The `apicm` login obtains a token that expires after eight hours. If the token expires while `apicm archive:push` is running, `apicm` generates errors due to lack of the necessary access permissions. In this case, log in again and re-run `apicm archive:push`. The token is refreshed every time you log in. See also [Token expiration in Migration troubleshooting](#)
- An alternative method of viewing endpoints is to run `kubectl get ingress` and get the values from the output. See [kubectl example](#).

10. When migration finishes, complete the following steps:

- If a user in a Local User Registry is owner of a Consumer Organization, the user must reset their password.
- If applicable, reapply any customized configurations from your v5 deployment. See [Customizations to recreate](#).
- Optional: If necessary, switch the provider organization ownership to the identity that owned it in v5. This step might be necessary if you created and migrated the provider organization using a different identity (such as administrative account) than the user identity of the v5 provider org. Note that the UI does not provide this function, but you can use the API `/orgs/{org}/transfer-owner` to switch ownership.
- If Stripe billing integrations were included in your migration, you must now configure the Stripe integrations in your Developer Portal sites; see [Configuring Stripe in the Developer Portal](#) for detailed instructions. For more information about the billing feature in v10, see [Monetizing your Products](#).

---

## Logging in to a management server with an OIDC registry

Logging in with an OIDC registry involves several steps, in which you obtain a temporary token and pass it to the toolkit to ensure it can communicate securely with the management server.

Attention: If you want to log in using a non-OIDC registry, see the login steps in [Migration steps without PDUR](#) or [Migration steps with PDUR](#). To log in, complete the following steps:

1. Enter the following command on the command line:

```
apicm login --server mgmt_endpoint_url --sso
```

where `mgmt_endpoint_url` is the endpoint URL. When you log in with a Cloud admin or Provider organization, specify one of the following URLs:

- platform API endpoint URL
- management API manager URL

These endpoint URLs are configured during the installation of API Connect, as described in [Installing the Management subsystem into a Kubernetes environment](#) and [Deploying the Management subsystem in a VMware environment](#). If you have access to the Cloud Manager user interface, you can view the configured endpoint URLs as described in [Viewing platform and UI endpoints](#), ignoring any segments at the end of the displayed URLs. If you are not sure of the endpoint URL, ask your administrator.

2. When the toolkit prompts for the context, type `provider`:

```
Context? provider
```

3. The server responds with the following message:

```
Please copy and paste the url https://mgmt_endpoint_url/auth/manager/sign-in/?from=TOOLKIT to a browser to start the authentication process.
Do you want to open the url in default browser? [y/n]: y
```

Take one of the following actions:

- If you want to use your default browser to log in to API Manager, type `y` and press Enter. The API Manager sign-in page opens in a new browser tab.
- If you don't use to use your default browser, type `n` and press Enter.
  - Copy the URL from the command window.
  - Open a browser, paste the URL, and press Enter. The API Manager sign-in page displays.

4. On the API Manager sign-in page, select the OIDC registry and then log in to API Connect. After you log in, API Connect displays the "You are authenticated!" message and provides a temporary token. Copy the token.
5. Return to the command window. Paste the token at the `API Key?` prompt and press Enter.

When the token is validated and you are successfully logged in to the toolkit, the following message displays:

```
Logged into mgmt_endpoint_url successfully
```

---

## Converting APIs to DataPower API Gateway

You can convert API Connect v5-compatible APIs to v10 DataPower® API Gateway.

### Before you begin

Use the migration utility command `apicm archive:port-to-apigw` to convert v5-compatible APIs to DataPower API Gateway. For instructions on installing the migration utility, see [Installing the migration utility](#).

You can convert your API definitions as part of the migration workflow, or as a separate task. Use these instructions to convert YAML source files separately. For instructions on converting APIs during migration, see [Migration steps without PDUR](#).

The following steps are required before converting v5-compatible APIs to DataPower API Gateway. For detailed instructions on all of these tasks, see [Migration steps without PDUR](#).

- Download the latest IBM API Connect® Migration Utility (AMU) from [IBM® Fix Central](#). The migration utility requires Installer Version 5.0.0 or later. However, if you intend to use it to convert APIs, use Installer Version 7.0.1 or later.
- Ensure that the AMU version that you are using is supported by your versions of DataPower API Gateway and API Connect. For more information, see [Configuring migration options for DataPower API Gateway](#).
- Converting APIs using the migration utility requires Node version 18.17.1. Ensure that you have that version installed.
- Run `archive:unpack` to unpack the V5 dbextract into v5-compatible artifacts.
- Map v5 object names to v10 object names. If the names are preserved exactly between the v5 and v10 environments, this step can be skipped. The gateway service must be mapped to a DataPower API Gateway. For more information, see [Migration mapping files](#).

For information about the differences between gateway types available in API Connect, see [API Connect gateway types](#).

### About this task

The migration utility command `apicm archive:port-to-apigw` provides assistance in converting v5-compatible APIs to DataPower API Gateway, rather than full migration. It generates an output API that is supported on API Gateway v10.

The conversion process annotates the YAML file and logs with information regarding the artifacts that it attempts to convert. For cases where additional information is available or further action is required, the YAML input file and logs contain links to documentation.

Because of differences between the programming models of v5-compatible APIs and DataPower API Gateway APIs, the command might not be able to convert all items. You might need to port these items to the API Gateway. In these cases, messages generated by the `apicm archive:port-to-apigw` command provide links to documentation that provides more details about changes you must make for these items to be compatible with DataPower API Gateway.

## Procedure

Run the conversion command from the directory where you downloaded the utility.

Command syntax:

```
./apicm archive:port-to-apigw path
```

The path is the full path to the YAML file or directory. You can specify to convert a single YAML file or the contents of an entire directory.

The migration utility produces an output YAML file with the results of the conversion process.

## What to do next

Review the output YAML file and make any necessary changes to ensure that your APIs are compatible with DataPower API Gateway. For more information, see [Reviewing APIs converted to DataPower API Gateway](#).

Push the converted YAML files to the v10 deployment. For details, see [Migration steps without PDUR](#).

## Configuring migration options for DataPower API Gateway

If you are converting APIs to DataPower API Gateway, you can accept the default migration configuration or specify configuration parameters.

You can specify migration configuration parameters using either a YAML configuration file or from the command line, but not both. The `config.yml` file must be located in the same directory as the migration utility. You can specify to use this file when the migration utility is invoked using the `apicm` command with the `port-to-apigw` option, as in the following example.

```
./apicm archive:port-to-apigw <path> --use-config-file=true
```

If use of the configuration file is not specified, the migration utility uses parameters specified on the command line. If no parameters are specified, then the default parameters are used.

Some of these `port-to-apigw` configuration parameters depend upon a specific DataPower implementation to function. See the information in the following table to ensure that you have the correct version of DataPower for the parameters you want to use.

Table 1. Migration configuration options

| Parameter                              | Valid values                                                                                                        | DataPower version required                                                                                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>activity-log-policy</code>       | <code>activity-log_1.5.0</code> , <code>set-variable_2.0.0</code>                                                   | DP 10.0.4.0+                                                                                                   | Defines how to process the v5 activity-log policy when the migration utility rewrites the API assembly. The default value is <code>activity-log_1.5.0</code> , which changes the activity-log version to 1.5.0. When set to <code>set-variable_2.0.0</code> , the migration utility sets context variables based upon the value of the activity-log content and error-content configuration values.                                                                                                                                                                                                                                                                                                                                              |
| <code>chunked-uploads</code>           | <code>true</code> / <code>false</code><br>The default value is <code>false</code> .                                 | For <code>invoke_1.5.0</code> :<br>10.0.1.1+,<br>10.0.2.0+<br><br>For <code>invoke_2.0.0</code> :<br>10.0.0.0+ | If an <code>invoke_1.5.0</code> or <code>proxy_1.5.0</code> policy exists in an assembly in the API, then set <code>chunked-uploads</code> to <code>true</code> to enable chunked encoding for HTTP 1.1 requests to the target server.<br><br>If an <code>invoke_2.0.0</code> policy exists in an assembly after the migration, then set <code>chunked-uploads</code> to <code>false</code> explicitly for the policy, as the default value for <code>chunked-uploads</code> in API Gateway is <code>true</code> .                                                                                                                                                                                                                               |
| <code>client-id-header-override</code> | String. The string cannot contain spaces. Follow the guidelines for <code>apiKey name</code> :<br>No default value. | 10.0.0.0+                                                                                                      | Override the existing <code>X-IBM-Client-Id</code> name value with the new value. In API Gateway, <code>x-key-type</code> makes it possible to define custom naming for <code>client_id</code> and <code>client_secret</code> .<br><br>Example:<br><pre>--client-id-header-override="acme-foo"</pre><br>In this example, the <code>securityDefinitions</code> property is changed from:<br><pre>foo:   type: apiKey   name: X-IBM-Client-Id   in: header</pre><br>to the following:<br><pre>foo:   type: apiKey   name: acme-foo   in: header   x-key-type: client_id</pre><br>Calls to this API must supply <code>acme-foo</code> :<br><client-id> in the request header instead of the original <code>X-IBM-Client-Id</code> :<br><client-id>. |

| Parameter                                  | Valid values                                                                                                        | DataPower version required | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------------------------------|---------------------------------------------------------------------------------------------------------------------|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>client-id-query-override</code>      | String. The string cannot contain spaces. Follow the guidelines for <code>apiKey name</code> .<br>No default value. | 10.0.0.0+                  | Override the existing <code>client_id</code> name value with the new value. In API Gateway, <code>x-key-type</code> makes it possible to define custom naming for <code>client_id</code> and <code>client_secret</code> .<br><br>Example:<br><pre>--client-id-query-override="acme-foo"</pre><br>In this example, the <code>securityDefinitions</code> property is changed from:<br><pre>foo:   type: apiKey   name: client_id   in: query</pre><br>to the following:<br><pre>foo:   type: apiKey   name: acme-foo   in: query   x-key-type: client_id</pre><br>Calls to this API must supply <code>?acme-foo=&lt;client-id&gt;</code> in the query parameter instead of the original <code>?client_id=&lt;client-id&gt;</code> .                                                 |
| <code>client-secret-header-override</code> | String. The string cannot contain spaces. Follow the guidelines for <code>apiKey name</code> .<br>No default value. | 10.0.0.0+                  | Override the existing <code>X-IBM-Client-Secret</code> name value with the new value. In API Gateway, <code>x-key-type</code> makes it possible to define custom naming for <code>client_id</code> and <code>client_secret</code> .<br><br>Example:<br><pre>--client-secret-header-override="acme-foo"</pre><br>In this example, the <code>securityDefinitions</code> property is changed from:<br><pre>foo:   type: apiKey   name: X-IBM-Client-Secret   in: header</pre><br>to the following:<br><pre>foo:   type: apiKey   name: acme-foo   in: header   x-key-type: client_secret</pre><br>Calls to this API must supply <code>acme-foo: &lt;client-secret&gt;</code> in the request header instead of the original <code>X-IBM-Client-Secret: &lt;client-secret&gt;</code> . |
| <code>client-secret-query-override</code>  | String. The string cannot contain spaces. Follow the guidelines for <code>apiKey name</code> .<br>No default value. | 10.0.0.0+                  | Override the existing <code>client_secret</code> name value with the new value. In API Gateway, <code>x-key-type</code> makes it possible to define custom naming for <code>client_id</code> and <code>client_secret</code> .<br><br>Example:<br><pre>--client-secret-query-override="acme-foo"</pre><br>In this example, the <code>securityDefinitions</code> property is changed from:<br><pre>foo:   type: apiKey   name: client_secret   in: query</pre><br>to the following:<br><pre>foo:   type: apiKey   name: acme-foo   in: query   x-key-type: client_secret</pre><br>Calls to this API must supply <code>?acme-foo=&lt;client-secret&gt;</code> in the query parameter instead of the original <code>?client_secret=&lt;client-secret&gt;</code> .                     |
| <code>copy-id-headers-to-message</code>    | <code>true</code> / <code>false</code><br>The default value is <code>false</code> .                                 | 10.0.1.1+,<br>10.0.2.0+    | When set to <code>true</code> , the <code>client-id</code> header is copied from <code>request.headers</code> to <code>message.headers</code> to be passed to the backend server. Set this parameter to <code>true</code> if your invoke backend services expect the <code>client-id</code> to be passed through.                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

| Parameter                         | Valid values                                                                                                                                                                                                                                                                                                                   | DataPower version required                                                                                                                                                                                                                                                                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>custom-policies-scope</b>      | <b>catalog / global</b><br>The default value is <b>catalog</b> .                                                                                                                                                                                                                                                               | 10.0.0.0+                                                                                                                                                                                                                                                                                   | <p>When set to <b>catalog</b>, the original v5 or v5-compatible catalog that contains the custom policy is staged for pushing to the target catalog and organization in API Gateway.</p> <p>When set to <b>global</b>, the custom policy is staged for pushing to API Gateway and is visible to all organizations and catalogs within that gateway.</p> <p><b>archive:port-to-apigw</b> sets up the custom policies for pushing to API Gateway, but does not do the pushing. If <b>global</b> scope is used, <b>archive:push</b> of the admin organization will push all custom policies associated with the configured gateways. If <b>catalog</b> scope is used, then the <b>archive:push</b> of the admin organization will initialize the custom policies on the gateway. A subsequent <b>archive:push</b> of the provider organization is needed to advertise the custom policies in the catalogs contained within the provider organization.</p> <p>Note: Potential conflicts between custom policies of the same name and version are resolved according to the following rules:</p> <ul style="list-style-type: none"> <li>When set to <b>global</b> and two policies under the same gateway service have the same name, version, and md5 hash value, then the duplicates are discarded and a message is generated.</li> <li>When set to <b>global</b> and two policies under the same gateway service have the same name, version, and a differing md5 hash value, then global scope cannot be applied. Catalog scope is used and a warning is generated.</li> <li>When set to <b>catalog</b>, <b>&lt;orgname&gt;_&lt;catname&gt;</b> is appended to the zip filename to maintain uniqueness.</li> </ul> |
| <b>deploy-policies</b>            | Any combination of the following values:<br><b>gatewayscript_1.0.0,if_1.5.0, invoke_1.5.0, proxy_1.5.0, redact_1.5.0, switch_1.5.0, validate-username-token_1.0.0, xslt_1.0.0</b><br>The default values are: <b>if_1.5.0, invoke_1.5.0, proxy_1.5.0, redact_1.5.0, switch_1.5.0, validate-username-token_1.0.0, xslt_1.0.0</b> | <p>For <b>invoke_1.5.0, proxy_1.5.0, gatewayscript_1.0.0</b>, and <b>xslt_1.0.0</b>: 10.0.0.0+</p> <p>For <b>if_1.5.0</b> and <b>switch_1.5.0</b>: 10.0.1.1+, 10.0.2.0+</p> <p>For <b>redact_1.5.0</b>: 10.0.3.0+</p> <p>For <b>validate-username-token_1.0.0</b>: 10.0.1.3+, 10.0.2.0+</p> | <p>Defines which backwards-compatible policies should be advertised by DataPower to API Manager.</p> <p>Note: Any legacy policies that are specified in the <b>config.yml</b> file or by using the <b>apicm</b> command must also be specified by using <b>deploy-policies</b>. If these policies are not specified by using <b>deploy-policies</b>, an error is returned.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>emulate-v4-plan-rate-limit</b> | <b>true / false</b><br>The default value is <b>false</b> .                                                                                                                                                                                                                                                                     | 10.0.1.2+, 10.0.2.0+                                                                                                                                                                                                                                                                        | Determines whether all APIs emulate the v4 plan rate limit. If set to <b>true</b> , then API operations do not share the rate limit. This setting matches the behavior of the <b>x-ibm-gateway-emulate-v4-plan-rate-limit</b> v5 option.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>enable-api-logging</b>         | <b>true / false</b><br>The default value is <b>true</b> .                                                                                                                                                                                                                                                                      | 10.0.0.0+                                                                                                                                                                                                                                                                                   | Log all changes that the migration utility makes when rewriting APIs. These changes are logged as comments in the API YAML file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>enforce-required-params</b>    | <b>true / false</b><br>The default value is <b>false</b> .                                                                                                                                                                                                                                                                     | 10.0.1.1+, 10.0.2.0+                                                                                                                                                                                                                                                                        | When set to <b>true</b> , required parameters are enforced, which is the default behavior in API Gateway. When set to <b>false</b> , required parameters are not enforced, even if the required parameter checkbox selected.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>gatewayscript-policy</b>       | One of the following values:<br><b>gatewayscript_1.0.0, gatewayscript_2.0.0</b><br>The default value is <b>gatewayscript_2.0.0</b> .                                                                                                                                                                                           | 10.0.0.0+                                                                                                                                                                                                                                                                                   | <p>Defines which gatewayscript policy the migration utility uses when rewriting the API assembly.</p> <p><b>gatewayscript_1.0.0</b>: A user-defined policy created by the migration utility. This policy uses the v5 MPGW processing rule. Use <b>gatewayscript_1.0.0</b> if the gatewayscript policy is dependent on v5 MPGW capabilities, such as load balancer groups or XML Manager settings.</p> <p><b>gatewayscript_2.0.0</b>: The migration utility rewrites the existing API to gatewayscript 2.0.0. This policy supports the <b>header-metadata</b> module for compatibility with the DataPower® Gateway (v5 compatible).</p> <p>Specifying <b>gatewayscript_2.0.0</b> can cause migration errors or run-time errors. Use <b>gatewayscript_2.0.0</b> if you are able to review and verify all changes made by the migration utility in the API YAML file.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>if-policy</b>                  | One of the following values:<br><b>if_1.5.0, switch_2.0.0</b><br>The default value is <b>if_1.5.0</b> .                                                                                                                                                                                                                        | <p>For <b>if_1.5.0</b>: 10.0.1.1+, 10.0.2.0+</p> <p>For <b>switch_2.0.0</b>: 10.0.0.0+</p>                                                                                                                                                                                                  | <p>Defines which <b>if</b> policy the migration utility uses when rewriting the API assembly.</p> <p><b>if_1.5.0</b>: The migration utility rewrites the existing API to if 1.5.0.</p> <p><b>switch_2.0.0</b>: The migration utility rewrites the existing API to switch 2.0.0. Specifying <b>switch_2.0.0</b> can cause migration errors or run-time errors. Use <b>switch_2.0.0</b> if you are able to review and verify all changes made by the migration utility in the API YAML file.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

| Parameter                             | Valid values                                                                                                               | DataPower version required                                                                         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>invoke-policy</b>                  | One of the following values:<br><b>invoke_1.5.0</b> ,<br><b>invoke_2.0.0</b><br>The default value is <b>invoke_1.5.0</b> . | 10.0.0.0+                                                                                          | Defines which invoke policy the migration utility uses when rewriting the API assembly.<br><br><b>invoke_1.5.0</b> : The migration utility rewrites the existing API to invoke 1.5.0.<br><br><b>invoke_2.0.0</b> : The migration utility rewrites the existing API to invoke 2.0.0. Specifying <b>invoke_2.0.0</b> can cause migration errors or run-time errors. Use <b>invoke_2.0.0</b> if you are able to review and verify all changes made by the migration utility in the API YAML file.                                                                                                                            |
| <b>no-rename</b>                      | <b>true</b> / <b>false</b><br>The default value is <b>false</b> .                                                          | 10.0.0.0+                                                                                          | Defines whether to append the porting suffix to artifact file versions or names or within artifacts. When set to <b>false</b> , the files are renamed using the porting suffix. When set to <b>true</b> , the original unpacked v5c files are overwritten with the ported API Gateway artifacts.<br><br>A <b>y/n</b> prompt appears when using this option. To bypass this prompt, use the <b>--accept-no-rename-deletion</b> option.                                                                                                                                                                                     |
| <b>no-retitle</b>                     | <b>true</b> / <b>false</b><br>The default value is <b>false</b> .                                                          | 10.0.0.0+                                                                                          | Defines whether to append the porting suffix to the <b>title</b> of the migrated API, product, or OAuth provider. Set this option to <b>true</b> if you want the <b>title</b> of the migrated API, product, or OAuth provider to match the original <b>title</b> in the v5-compatible YAML file.                                                                                                                                                                                                                                                                                                                          |
| <b>optimize-gws</b>                   | <b>true</b> / <b>false</b><br>The default value is <b>false</b> .                                                          | 10.0.0.0+                                                                                          | When set to <b>true</b> , gatewayscript policies are modified to map v5 functions to API Gateway functions.<br><br>Setting <b>optimize-gws</b> to <b>true</b> is valid only for the <b>gatewayscript_2.0.0</b> policy. If it is set to <b>true</b> for <b>gatewayscript_1.0.0</b> , an error is generated.<br><br>Note: Setting <b>optimize-gws</b> to <b>true</b> can result in performance enhancements. However, the modifications to gatewayscript policies by the migration utility can cause problems. When using this setting, review all gatewayscript policy modifications after migration.                      |
| <b>porting-suffix</b>                 | A string containing alphanumeric, dash, underscore and dot characters.<br>The default value is <b>-apigw</b> .             | 10.0.0.0+                                                                                          | Defines the suffix to be used when renaming APIs, Products, OAuth Providers, and other ported artifacts. The suffix is appended to the version on APIs, Products and other ported artifacts that have versions, and onto the name for OAuth Providers and other ported artifacts that do not have versions. If <b>porting-suffix</b> is not specified then a value of <b>-apigw</b> will be used. The porting suffix can contain alphanumeric, dash, underscore and dot characters. Prior to AMU 10.0.5.4 the porting suffix could not be overridden and <b>-apigw</b> was appended to the name for all ported artifacts. |
| <b>proxy-policy</b>                   | One of the following values:<br><b>proxy_1.5.0</b> ,<br><b>invoke_2.0.0</b><br>The default value is <b>proxy_1.5.0</b> .   | 10.0.0.0+                                                                                          | Defines which proxy policy the migration utility uses when rewriting the API assembly.<br><br><b>proxy_1.5.0</b> : The migration utility rewrites the existing API to <b>proxy_1.5.0</b> .<br><br><b>invoke_2.0.0</b> : The migration utility converts the proxy policy to invoke 2.0.0. Specifying <b>invoke_2.0.0</b> can cause migration errors or run-time errors. Use <b>invoke_2.0.0</b> if you are able to review and verify all changes made by the migration utility in the API YAML file.                                                                                                                       |
| <b>redact-policy</b>                  | One of the following values:<br><b>redact_1.5.0</b> ,<br><b>redact_2.0.0</b><br>The default value is <b>redact_1.5.0</b> . | 10.0.3.0+                                                                                          | Defines which redact policy the migration utility uses when rewriting the API assembly.<br><br><b>redact_1.5.0</b> : The migration utility rewrites the existing API to <b>redact_1.5.0</b> .<br><br><b>redact_2.0.0</b> : The migration utility rewrites the existing API to <b>redact_2.0.0</b> . Specifying <b>redact_2.0.0</b> can cause migration errors or run-time errors, including changes to what is redacted, particularly with redactions meant for JSON payloads. Use <b>redact_2.0.0</b> if you are able to review and verify all changes made by the migration utility in the API YAML file.               |
| <b>repair-wsdl-apis</b>               | <b>true</b> / <b>false</b><br>The default value is <b>true</b> .                                                           | 10.0.1.1+,<br>10.0.2.0+                                                                            | If an API ported from V5 is created from WSDL, the WSDL is read and checked for accuracy. If <b>repair-wsdl-apis</b> is set to <b>true</b> , then the API and WSDL are automatically repaired. For example, if the WSDL ZIP file contains unnecessary files, they are automatically removed if <b>repair-wsdl-apis</b> is <b>true</b> .                                                                                                                                                                                                                                                                                   |
| <b>return-v5-responses</b>            | <b>true</b> / <b>false</b><br>The default value is <b>false</b> .                                                          | 10.0.1.1+,<br>10.0.2.0+                                                                            | When set to <b>true</b> , OAuth-related messages are printed in the v5 response format. When set to <b>false</b> , OAuth-related response messages are printed in API Gateway response format.                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>switch-policy</b>                  | <b>switch_1.5.0</b> ,<br><b>switch_2.0.0</b><br>The default value is <b>switch_1.5.0</b> .                                 | For <b>switch_1.5.0</b> :<br>10.0.1.1+,<br>10.0.2.0+<br><br>For <b>switch_2.0.0</b> :<br>10.0.0.0+ | Defines which switch policy the migration utility uses when rewriting the API assembly.<br><br><b>switch_1.5.0</b> : The migration utility rewrites the existing API to switch 1.5.0.<br><br><b>switch_2.0.0</b> : The migration utility rewrites the existing API to switch 2.0.0. Specifying <b>switch_2.0.0</b> can cause migration errors or run-time errors. Use <b>switch_2.0.0</b> if you are able to review and verify all changes made by the migration utility in the API YAML file.                                                                                                                            |
| <b>use-config-file</b>                | <b>true</b> / <b>false</b><br>The default value is <b>false</b> .                                                          | 10.0.0.0+                                                                                          | Use the <b>config.yml</b> file that is in the same directory as the <b>apicm</b> binary file that is being executed. If this file is used, configuration options cannot be set using the command line. The purpose of the config file is to specify the command inputs given on this page, but in an easy-to-read file format (YAML).                                                                                                                                                                                                                                                                                     |
| <b>v5-request-headers</b>             | <b>true</b> / <b>false</b><br>The default value is <b>true</b> .                                                           | 10.0.1.2+,<br>10.0.2.0+                                                                            | When set to <b>true</b> , the following v5-compatible request headers are set: <b>Via</b> , <b>X-Global-Transaction-ID</b> , <b>X-Client-IP</b> . When set to <b>false</b> , these request headers are not set.                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>validate-username-token-policy</b> | <b>validate-username-token_1.0.0</b><br>The default value is empty.                                                        | 10.0.1.3+,<br>10.0.2.0+                                                                            | Defines whether or not to use the <b>validate-username-token</b> v5 emulation policy when the migration utility rewrites the API assembly. When empty, the migration utility takes no action on the policy.                                                                                                                                                                                                                                                                                                                                                                                                               |

| Parameter                | Valid values                                                                                                                           | DataPower version required | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>xslt-policy</code> | One of the following values:<br><code>xslt_1.0.0</code> ,<br><code>xslt_2.0.0</code><br>The default value is <code>xslt_1.0.0</code> . | 10.0.0.0+                  | Defines which gatewayscript policy the migration utility uses when rewriting the API assembly.<br><br><code>xslt_1.0.0</code> : A user-defined policy created by the migration utility. This policy uses the v5 MPGW processing rule. Use <code>xslt_1.0.0</code> if the xslt policy is dependent on v5 MPGW capabilities, such as load balancer groups or XML Manager settings.<br><br><code>xslt_2.0.0</code> : The migration utility rewrites the existing API to xslt 2.0.0. Specifying <code>xslt_2.0.0</code> can cause migration errors or run-time errors. Use <code>xslt_2.0.0</code> if you are able to review and verify all changes made by the migration utility in the API YAML file. |

## Example migration configurations

The following examples show configuration options using the `config.yml` file or the `apicm` command.

- The following example uses the `config.yml` file to specify the migration configuration.

```
custom-policies-scope: global
emulate-v4-plan-rate-limit: false
deploy-policies:
- proxy_1.5.0
- redact_1.5.0
- invoke_1.5.0
- xslt_2.0.0
- gatewayscript_1.0.0
- switch_2.0.0
rewrite-apis:
enable-api-logging: false
proxy-policy: proxy_1.5.0
redact-policy: redact_1.5.0
invoke-policy: invoke_1.5.0
xslt-policy: xslt_2.0.0
gatewayscript-policy: gatewayscript_1.0.0
if-policy: switch_2.0.0
switch-policy: switch_2.0.0
optimize-gws: false
no-retitle: true
```

Use the `use-config-file` parameter when running the `apicm` command to specify that the `config.yml` file is used for migration configuration values.

```
./apicm archive:port-to-apigw <path> --use-config-file=true
```

- The following example uses the `apicm` command to specify the same settings as in the previous example.

```
./apicm archive:port-to-apigw <path> --custom-policies-scope: global --emulate-v4-plan-rate-limit: false --deploy-policies="proxy_1.5.0,redact_1.5.0,invoke_1.5.0,xslt_2.0.0,gatewayscript_1.0.0,switch_2.0.0" --enable-api-logging=false --proxy-policy="proxy_1.5.0" --redact-policy="redact_1.5.0" --invoke-policy="invoke_1.5.0" --xslt-policy="xslt_2.0.0" --gatewayscript-policy="gatewayscript_1.0.0" --if-policy="switch_2.0.0" --switch-policy="switch_2.0.0" --optimize-gws=true --no-retitle=true
```

- The following example uses the `apicm` command to specify that the default migration configuration values are used.

```
./apicm archive:port-to-apigw <path>
```

## Reviewing APIs converted to DataPower API Gateway

The migration utility command `apicm archive:port-to-apigw`, which assists in the conversion of v5 APIs for use with the v10 DataPower® API Gateway, provides draft output YAML files containing the results of the conversion process.

### Procedure

- Review the entire output YAML files for conformance, performance, and errors.  
The conversion process annotates the YAML file and logs with information regarding the artifacts that are converted. For cases where additional information is available or further action is required, the YAML file and logs contain links to documentation.
- Make any necessary changes to port APIs to DataPower API Gateway.
  - [Messages generated during conversion of APIs to DataPower API Gateway](#)  
Reference information about the messages generated by the migration utility command `apicm archive:port-to-apigw`.
  - [Parse actions added to the API assembly after migration](#)  
The migration utility command `apicm archive:port-to-apigw` adds parse actions to the API assembly depending on the policies specified in your migration configuration.
  - [Fine-tuning APIs after migration](#)  
After running the migration utility command `apicm archive:port-to-apigw`, review the converted APIs to determine if any of the changes made by the migration utility can be modified to enhance performance and usability.
  - [Frequently asked questions about migrating APIs to API Gateway](#)  
Answers to some of the most frequently asked questions about migrating APIs to API Gateway.

### Related information



- [Migrating a Version 5 deployment](#)

---

## Messages generated during conversion of APIs to DataPower API Gateway

Reference information about the messages generated by the migration utility command `apicm archive:port-to-apigw`.

[Activity log messages](#)

[Entire API/General messages](#)

---

### Policies

- [GatewayScript policy messages](#)
- [Invoke policy messages](#)
- [JSON-to-XML policy messages](#)
- [JWT-Generate policy messages](#)
- [LTPA-Generate policy messages](#)
- [Map policy messages](#)
- [Proxy policy messages](#)
- [Redact policy messages](#)
- [Set-Variable policy messages](#)
- [Switch policy messages](#)
- [Throw policy messages](#)
- [Validate policy messages](#)
- [WSDL messages](#)
- [XML-to-JSON policy messages](#)
- [XSLT policy messages](#)

[Migration utility messages](#)

[Miscellaneous messages](#)

---

### Referenced objects

- [OAuth2 messages](#)
- [TLS-Profile messages](#)

---

### Activity log messages

- [ACT06](#)
- [ACT07](#)
- [ACT09](#)

**ACT06** `activity-log` policy set to `activity-log\_1.5.0`, converting `activity-log` to `activity-log\_1.5.0`.

#### Explanation

See message text.

#### Severity

info

**ACT07** `activity-log` policy set to `set-variable\_2.0.0`, converting `activity-log` to `set-variable\_2.0.0`.

#### Explanation

The `activity-log` success and error values are saved into context variables. A subsequent log policy will reference these variables, which will generate the analytics log data based upon the desired log level.

#### Severity

info

**ACT09** For `activity-log` policy set to `set-variable\_2.0.0`, added a `switch` policy with `log` policies into the API finally clause to utilize the `activity-log` variables that were set.

#### Explanation

A `switch` policy determines if the analytics log level should be based upon the success of the API or on error `activity-log` settings. The context variable set in a prior `set-variable` policy is used as the `log` policy log level to gather the appropriate level of data in the analytics log.

#### Severity

info

## Entire API/General messages

---

- [API01](#)
- [API02](#)
- [API03](#)
- [API04](#)
- [API05](#)
- [API06](#)
- [API07](#)
- [API08](#)
- [API09](#)
- [API10](#)
- [API11](#)
- [API12](#)
- [API13](#)
- [API14](#)
- [API15](#)
- [API16](#)
- [API17](#)
- [API18](#)
- [API19](#)
- [API20](#)
- [API21](#)
- [API22](#)
- [API23](#)
- [API24](#)
- [API25](#)
- [API26](#)
- [API27](#)
- [API28](#)
- [API29](#)
- [API30](#)
- [API31](#)
- [API32](#)
- [API33](#)
- [API34](#)
- [API35](#)
- [API36](#)
- [API37](#)
- [API38](#)
- [API39](#)
- [API40](#)
- [API41](#)
- [API42](#)
- [API43](#)
- [API44](#)
- [API45](#)
- [API46](#)
- [API47](#)
- [API48](#)
- [API49](#)
- [API51](#)
- [API52](#)
- [API53](#)
- [API54](#)
- [API55](#)
- [API56](#)
- [API57](#)
- [API58](#)
- [API59](#)
- [API60](#)
- [API61](#)
- [API62](#)
- [API63](#)
- [API64](#)
- [API65](#)
- [API66](#)
- [API67](#)

**API01** The specified API YAML input file is of gateway type `datapower-api-gateway`. This utility accepts only v5-compatible APIs of gateway type `datapower-gateway`.

### Explanation

The input API YAML file must be of gateway type `datapower-gateway`. If the gateway type is `datapower-api-gateway`, the API does not need to be converted. Any other gateway type is not supported.

### Response

Review your input file to confirm that it is v5 compatible.

### Severity

error

**API02 Input API policy is already version 2.0.0. This version is reserved for API Gateway policies only. Skipping conversion of this policy.**

### Explanation

The input API YAML file must be of gateway type `datapower-gateway`. Every policy within that API must be version 1.0.0. Version 2.0.0 policies are reserved for the API Gateway, which should be included only in APIs with gateway type `datapower-api-gateway`. No conversion is performed for this policy.

### Severity

error

**API03 Suffix ``%s`` appended to API ``title`` and ``x-ibm-name``.**

### Explanation

As the v5 and API Gateway policy can reside in the same API Manager, a suffix can be provided to ensure a unique name of the API Gateway version of the v5 API.

### Response

Consider incrementing the API version to accommodate converting to the API Gateway.

### Severity

info

**API04 Added missing ``/`` to the front of the basepath.**

### Explanation

A v5 API with a `basepath` property value that does not start with a `/` will have the character `/` added at the beginning of the `basepath` property of the API Gateway.

### Severity

info

**API05 Trailing slashes (``/``) removed from the basepath to conform to the API Gateway specification.**

### Explanation

A v5 API with a `basepath` property value that is not a `/` but ends with the `/` character will have the ending `/` character removed in the `basepath` property of the API Gateway.

### Response

Calls to the API might need to be updated to conform to the new basepath.

### Severity

warn

**API06 Changed gateway type to `datapower-api-gateway``.**

### Explanation

See message text.

### Severity

info

**API07 Moved ``assembly request`` into ``assembly execute``.**

### Explanation

Older v5 APIs might include the `execute` property within an assembly `request` property. This `execute` property is moved to the API Gateway `assembly`` property.

### Severity

info

**API08 Moved ``assembly implementation`` into ``assembly execute``.**

### Explanation

Older v5 APIs might include the `execute` property within an assembly `implementation` property. This `execute` property is moved to the API Gateway `assembly` property.

### Severity

info

**API09 Moved `assembly response` into `assembly execute`.**

### Explanation

Older v5 APIs might have the `execute` property within an assembly `response` property. This `execute` property is moved to the API Gateway `assembly` property.

### Severity

info

**API10 Removed `x-ibm-gateway-invoke-v4-nomap-emulation` API property and replaced it with a `gatewayscript` policy to conform to the API Gateway specification.**

### Explanation

This v5 API property applies only to the `invoke` policy. In API Gateway, this property has been replaced by a `gatewayscript` policy that will modify the `invoke` policy response in a similar fashion.

### Severity

warn

**API11 The API property `x-ibm-gateway-sourcecode-resolve-apic-variables` does not exist in the API Gateway. The property has been removed from the YAML file.**

### Explanation

This v5 API property is not supported by the API gateway and has been removed. If the `x-ibm-gateway-map-resolve-apic-variables` property is not set, then the original value of this property will be set into the map option `mapResolveApicVariables` automatically.

### Severity

error

**API12 The API property `x-ibm-gateway-reencode-request-params-slash` does not exist in the API Gateway. The property has been removed from the YAML file.**

### Explanation

This v5 API property only applies to the `invoke` and `proxy` policies. In the API Gateway, the functionality of this property has been implemented as an option in the `invoke` policy.

This change was made for the following reasons.

- The option is now clearly visible in the `invoke` policy, making it easy to identify available options.
- Each use of the `invoke` policy can now have a different setting.

### Severity

error

**API13 The API property `x-ibm-gateway-framework-legacy-json-encoding` does not exist in the API Gateway. The property has been removed from the YAML file.**

### Explanation

This v5 API property is not supported by the API gateway and has been removed. If the value of the `x-ibm-gateway-framework-legacy-json-encoding` property is present, it will be used by the migration utility when migrating `xslt`, `gatewayscript`, and `map` policy source code to produce the same result as v5 would have produced based upon the property value.

### Severity

error

**API14 Converted v5-compatible policies to the API Gateway and updated policy versions to 2.0.0.**

### Explanation

See message text.

### Severity

info

**API15** The `x-ibm-gateway-api-enforce-response-limits` API property is enabled. Appended `parse` policy to the end of the assembly to parse the JSON response.

#### Explanation

The v5 API property `x-ibm-gateway-api-enforce-response-limits` ensures that a JSON response payload is within the defined JSON parser limits. This functionality is emulated in API Gateway by adding a `parse` policy at the end of the assembly.

#### Severity

info

**API16** Removed the `x-ibm-gateway-api-enforce-response-limits` API property and replaced it with a `parse` policy to conform to the API Gateway specification.

#### Explanation

The v5 API property `x-ibm-gateway-api-enforce-response-limits` ensures that a JSON response payload is within the defined JSON parser limits. This functionality is emulated in API Gateway by replacing it with a `parse` policy.

#### Severity

info

**API17** Setting `x-ibm-soap` to `%s`.

#### Explanation

The v5 API path does not have a `x-ibm-soap` property with a `soap-action` child property, but the schema referenced by this path is for a SOAP document. The `x-ibm-soap` property is added with a child property of `soap-action` into this path of the API gateway. The `soap-action` property will be determined from the schema encountered.

#### Severity

info

**API18** This `set-variable` policy is used to add the client identification header to the message headers for use by a subsequent `invoke` policy.

#### Explanation

Typically, the client identification header as defined by the security definition is `X-IBM-Client-Id`. This header will be removed from the message headers at the start of the API. A `set-variable` policy will add this header back into the message headers, as the v5 API specifies not to suppress this header from being sent by subsequent `invoke` policies.

#### Severity

info

**API19** Skipping appending of a `parse` policy for the `x-ibm-gateway-api-enforce-response-limits` API property, as this `parse` policy already exists at the end of the assembly and contains the same `input` and `output` values.

#### Explanation

See message text.

#### Severity

info

**API20** Added `parse` policy to the beginning of the assembly to parse incoming data.

#### Explanation

The v5-compatible implementation implicitly parses messages. However, by requiring explicit parsing for every assembly wherever parsing is required, the API Gateway provides greater control over when and how parsing is performed.

The `parse` policy is added in API Gateway for the following purposes.

- To explicitly control when parsing occurs.
- To determine which threat protection settings to apply to a parse.
- To determine which kinds of output to allow.
- To determine how to gather metadata from the parse.

By separating the parsing time from the processing time of the assembly, explicit parsing can also enhance performance.

API Gateway has no implicit parsing. The assembly must explicitly parse when parsing is desired, which ensures that the correct limits and input checks are done.

#### Response

You should retain the added `parse` policy for any of the following reasons.

- You need non-well-formed XML or JSON documents to abort and call the `catch`.

- The data is still streaming and you need to keep a copy of the data for various reasons.
- You need to use the payload data with another policy.

If you do not need the **parse** policy, remove it. If you retain the **parse** policy, consider the following questions.

- Which threat protection settings do you want to enforce with this **parse** policy?
- Does the use of this **parse** policy remove the need for an additional **parse** policy?
- Do you want to use the metadata from the parse, such as document size and complexity?

### Severity

info

**API21 Moved catalog specific `x-ibm-gateway-\*` API Property: `%s` under catalog: `%s` to API level for further processing.**

### Explanation

**x-ibm-gateway** catalog-level properties are not supported in API Gateway. The properties have been moved to the top-level properties section in the API for further processing by the migration utility.

### Severity

info

**API22 Skipped adding `parse` policy since an adjacent parse of the same properties was already added.**

### Explanation

The **invoke** policy might add a **parse** policy next to it in the assembly. If this **parse** policy is detected, another parse is not added because it would result in duplicate parses in the assembly. This prevention of duplicate parses applies only if both parses are found to have the same properties.

### Severity

info

**API23 Removed redundant `parse` policy from end of the assembly.**

### Explanation

The tool added a duplicate **parse** policy to the end of the assembly. The duplicate policy has been removed as it performs the same action as another **parse** policy in the assembly.

### Severity

info

**API24 Removed redundant `parse` policy from the beginning of the assembly.**

### Explanation

The tool added a duplicate **parse** policy to the beginning of the assembly. The duplicate policy has been removed as it performs the same action as another **parse** policy in the assembly.

### Severity

info

**API25 Removed unsupported `name` property from the Swagger info property.**

### Explanation

An empty **name** property under the **info** property is not supported in API Gateway. If an API is published with an empty **name** property, an error is generated. Therefore, the migration utility removes the empty **name** property if it is found.

### Severity

info

**API26 Removed empty `description` property from the Swagger info property.**

### Explanation

An empty **description** property is not supported in API Gateway. If an API is published with an empty **description** property, an error is generated. Therefore, the migration utility removes the empty **description** property if it is found.

### Severity

info

**API27 Removed empty `security` property from the Swagger file.**

### Explanation

An empty object `security` property is not supported in API Gateway. If an API is published with an empty object `security` property, an error is generated. Therefore, the migration utility removes the empty object `security` property if it is found.

### Severity

info

### API28 Moved `%s` from API properties to this policy option `%s` and set to [%s].

### Explanation

In the API Gateway, the functionality of this property has been implemented as a policy option.

### Severity

info

### API29 Preserved API Property `%s` in YAML.

### Explanation

The API property was preserved because there is a 1.0.0 or 1.5.0 policy that requires it.

### Severity

info

### API30 Removed API Property `%s` from YAML.

### Explanation

This property is obsolete and has been removed from the YAML file.

### Severity

info

### API31 The API property `x-ibm-gateway-custom-policy-with-gws-action` does not exist in the API Gateway. The property has been removed from the YAML file.

### Explanation

In v10, this property is not required to control the behavior of a custom policy that includes a GatewayScript action, as both `true` and `false` scenarios are handled as expected.

### Severity

error

### API32 The `$ref %s` was not found in the swagger.

### Explanation

A `$ref` could not be dereferenced because the reference content was not found in the v5 YAML.

### Response

Review the v5 file to confirm that the `$ref` references are valid and exist in the YAML.

### Severity

warn

### API33 Found usage of inline parameters: `%s` in `%s` policy.

### Explanation

API Gateway does not support inline parameters in the GatewayScript 2.0.0 or XSLT 2.0.0 policies.

### Response

Modify the policy to use context variables instead of inline parameters. The following tables show examples of inline parameters in v5 and how they can be rewritten to use context variables in API Gateway.

Table 1. GatewayScript 2.0.0 examples

| v5                                                                     | API Gateway                                                 |
|------------------------------------------------------------------------|-------------------------------------------------------------|
| If <code>test.data</code> is a variable:<br><code>\$(test.data)</code> | <code>context.get('test.data');</code>                      |
| If <code>test.body</code> is a message:<br><code>\$(test.body)</code>  | <code>context.getMessage('test').body;</code>               |
| <code>{test.example}</code>                                            | <code>context.get('request.parameters.test.example')</code> |

For more GatewayScript 2.0.0 examples, see [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway.](#)

Table 2. XSLT 2.0.0 example

| v5                                          | API Gateway                                                                                                             |
|---------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| <code>\$(request.parameters.example)</code> | <code>apigw:get-variable('request.parameters.example')</code><br>Note: Be sure to use the <code>apigw</code> namespace. |
| <code>{example}</code>                      | <code>apigw:get-variable('request.parameters.example')</code>                                                           |

For more XSLT 2.0.0 examples, see [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway.](#)

### Severity

error

**API34 Found `x-ibm-gateway-\*` catalog specific property: `%s` under catalog section: `%s`, within an unpublished API.**

### Explanation

The API was most likely found under the drafts folder of an organization. Because it is not published to a specific catalog, the migration utility cannot determine which API to apply the properties to.

### Response

Apply the `x-ibm-gateway-*` properties to the policies of the assembly manually before publishing.

### Severity

info

**API35 Found `x-ibm-gateway-\*` catalog specific property: `%s` under catalog section: `%s`. This API is published under catalog: `%s`**

### Explanation

The API is published to a specific catalog, and other catalog-specific `x-ibm-gateway-*` properties were found under other catalogs.

### Response

If you want to move this API to other catalogs, apply these properties manually to the policies in the assembly.

### Severity

info

**API36 `x-ibm-gateway-\*` API Property: `%s` found to have invalid value: `%s`.**

### Explanation

See message text.

### Response

Check the API to ensure that the API property has the correct value. For more information on the API property, see the appropriate section of the IBM API Connect Knowledge Center.

### Severity

error

**API37 JSON Schema contains an invalid reference.**

### Explanation

See message text.

### Response

Confirm that your schema definitions are valid. If you find errors, fix them.

### Severity

info

**API38 XSD generation ignoring references beyond limit set by property.**

### Explanation

XSD generation from JSON schema uses the `x-ibm-gateway-schema-definition-reference-limit` property value to determine how many recursions to go in circular references.

### Response

### Severity



info

**API39 An error occurred while generating the XSD for ` %s ` definition.**

**Explanation**

See message text.

**Response**

Confirm that your JSON schema is valid before continuing.

**Severity**

warn

**API40 Skipping JSON Schema definition: ` %s ` with multiple roots or not intended for XML data.**

**Explanation**

See message text.

**Severity**

info

**API41 Set ` chunked-uploads ` value to ` false ` to emulate v5 behavior.**

**Explanation**

For information about the Allow chunked uploads property in the Invoke policy, see [Invoke](#).

**Severity**

info

**API42 Input option ` chunked-uploads ` set to ` true `, which is the API Gateway ` chunked-uploads ` default. All invoke and proxy 1.5.0 and 2.0.0 policies will use this default value.**

**Explanation**

For information about the Allow chunked uploads property in the Invoke policy, see [Invoke](#).

**Severity**

info

**API43 Created ` compatibility: ` section for API-wide toggles set by migration utility.**

**Explanation**

The migration utility generates a compatibility section that allows DataPower to read in custom settings to for compatibility between v5 and API Gateway.

**Severity**

info

**API44 Set compatibility toggle ` %s ` to value: ` %s `.**

**Explanation**

See message text.

**Severity**

info

**API45 Compatibility toggle: ` %s ` set to value: ` %s `, which is the same as the API Gateway default.**

**Explanation**

The migration utility input specified to set a compatibility toggle to a value that is already the default. Therefore, no change was made in the API as the default will automatically be used.

**Severity**

info

**API46 Updated ` securityDefinition `: ` %s ` with new name: ` %s ` as specified in toggle input: ` %s `.**

**Explanation**

See message text.

### Severity

info

**API147 `application-authentication` was found to be set, adding `application-authentication-source` and `custom-header-name` defaults.**

### Explanation

In v5, `application-authentication-source` does not exist. In API Gateway, it is a new option that must be set if authentication is set. Therefore, the migration utility automatically sets `application-authentication-source` to its most-compatible default, and as a result also sets `custom-header-name`. If `application-authentication` is not set in the v5 API, the migration utility does not set `application-authentication-source` and `custom-header-name`.

### Severity

info

**API148 `%s` found in API definition `%s`, changing the value to a string null.**

### Explanation

See message text.

### Severity

info

**API149 Changed `\$(request.body\*)` to `\$(request.parsed\_body\*)` in: `%s` to use parsed copy of data.**

### Explanation

In the API Gateway, the `request.body` context is never parsed and cannot be evaluated. The parse policy added to the beginning of the assembly automatically copies `request.body` into `request.parsed_body` and parses that copy. The reference to `request.body` has been changed to use this new context.

### Severity

warn

**API151 Added missing path parameter definition for path `%s` operation `%s` parameter `%s`.**

### Explanation

See message text.

### Severity

info

**API152 Added missing path parameter reference for path `%s` operation `%s` parameter `%s`.**

### Explanation

See message text.

### Severity

info

**API153 A path parameter reference `%s` is not a path parameter.**

### Explanation

See message text.

### Severity

warn

**API154 A parameter reference `%s` is not defined.**

### Explanation

See message text.

### Severity

warn

**API155 Definition: %s is missing property 'Body' or 'properties', this is an invalid SOAP definition.**

### Explanation

The structure of the SOAP definition is invalid. For example, if the definition contains a SOAP fault that is not within the SOAP body, then the definition is invalid.

### Severity

error

**API56 Catalog specific API Property found without a corresponding default property: %s for catalog: %s, created empty string default property at the API level.**

### Explanation

See message text.

### Severity

info

**API57 Definition reference: %s is incorrect, and should be of form '#/definitions/foo'**

### Explanation

See message text.

### Response

For example, the following reference is invalid:

```
moreDetails:
 $ref: '#definitions/MoreDetails'
```

To make this a valid reference, use the following form:

```
moreDetails:
 $ref: '#/definitions/MoreDetails'
```

### Severity

warn

**API58 `securityDefinition` property encountered without a `security` property. An empty `security` property is added.**

### Explanation

See message text.

### Severity

info

**API59 The missing referenced definition `%s` was added to the swagger.**

### Explanation

See message text.

### Severity

info

**API60 The API property name `%s` is invalid and has been removed from the swagger.**

### Explanation

API properties with an invalid characters such as \$ and parentheses are ignored.

### Severity

info

**API61 Set buffering property to true.**

### Explanation

APIs will always have buffering enabled, as v5 did not support streaming.

### Severity

info

**API62 A query based security definition `%s` was present. A new security definition `%s` was created to match implicit v5 behavior.**

### Explanation

The security definitions in v5 for an API key that was specified as a query parameter were configured using the key names `client_id` and `client_secret`. However, this behavior implicitly provided the ability to also use query parameters `app_id` and `app_secret`. The specification of `app_id` and `app_secret` must be done explicitly in v10.

### Severity

info

**API63 A security set was created with security definition references `%s` to match implicit v5 behavior.**

### Explanation

The security definitions in v5 for an API key that was specified as a query parameter were configured using the key names `client_id` and `client_secret`. However, this behavior implicitly provided the ability to also use query parameters `app_id` and `app_secret`. The specification of `app_id` and `app_secret` must be done explicitly in v10.

### Severity

info

**API64 Suffix `%s` appended to API `x-ibm-name`.**

### Explanation

See message text.

### Severity

info

**API65 The referenced definition `%s` was changed to `%s`.**

### Explanation

The referenced definition contained a `/` character in the name and was not defined in the API. The `/` character has been removed to allow the API to publish during `archive:push`.

### Severity

info

**API66 Security has been removed as there are no associated security definitions.**

### Explanation

The API security is extraneous without associated security definitions.

### Severity

info

**API67 The API property `x-ibm-gateway-framework-preserve-escaped-reverse-solidus` does not exist in the API Gateway. The property has been removed from the YAML file.**

---

### Explanation

This v5 API property is not supported by the API gateway and has been removed. If the `x-ibm-gateway-framework-preserve-escaped-reverse-solidus` API property is present with a `true` value, it will be used by the migration utility when migrating `xslt`, `gatewayscript`, and `map` policy source code to produce the same result as v5 would have produced.

### Severity

info

## Policies

---

### GatewayScript policy messages

- [GWS02](#)
- [GWS04](#)
- [GWS05](#)
- [GWS06](#)
- [GWS07](#)
- [GWS08](#)
- [GWS09](#)
- [GWS10](#)
- [GWS11](#)
- [GWS12](#)

- [GWS18](#)
- [GWS19](#)
- [GWS20](#)
- [GWS21](#)
- [GWS22](#)
- [GWS24](#)
- [GWS25](#)
- [GWS26](#)

**GWS02 This `gatewayscript` policy was added before the newly converted `switch` policy in the assembly to assist with the condition logic.**

**Explanation**

A conditional policy from a v5 API is converted to a `switch` policy in the API Gateway. The v5 API uses GatewayScript for the `if` and `switch` policies, whereas the API Gateway condition syntax is JSONata based. These conditions have been moved into a `gatewayscript` policy that will execute immediately before the `switch` policy. The switch conditions will use results from the `gatewayscript` policy to execute the correct assembly path.

**Response**

Review the changes to ensure that the `switch` policy result will be consistent with the v5 API.

**Severity**

info

**GWS04 Found `gatewayscript` policy. Porting might be required.**

**Explanation**

Review the `gatewayscript` policy and make any necessary changes. The following list includes examples of some of the differences between `gatewayscript` code in v5-compatible APIs and API Gateway.

- Calls that reference the `request.parameters` context variable in v5-compatible APIs might not produce the same results in API Gateway. For example, consider the following call in a v5-compatible API.

```
apim.getvariable('request.parameters.org_name')
```

This call is not equivalent to the following call in an API Gateway API.

```
context.get('request.parameters.org_name')
```

The `context.get()` call in API Gateway supports multiple occurrences of the same parameter variable. Therefore, it returns structured JSON for retrieving parameters, as in the following example return.

```
"{"locations":["query"],"values":["sandbox-test-org"]}"
```

For this example, the `apim.getvariable` call should be changed to the following call in API Gateway.

```
((context.get('request.parameters.org_name') || {}).values || []).slice(-1)[0]
```

- Access to `header-metadata` is not supported in API Gateway. In some cases, the call can be edited to conform to the API Gateway specification. For example, the call `hm.current.remove("Accept-Encoding");` can be replaced in API Gateway with `context.message.header.remove("message.headers.Accept-Encoding")` or `context.clear("message.headers.Accept-Encoding")`. However, if you are using `header-metadata` to set the response message header before the `proxy` policy, you must add a `gatewayscript` policy after the `invoke` policy to set the header on the response message.
- In v5 and in v10 API Gateway, `apim.getvariable()` returns an XML node set if an XML node set was set in the variable being retrieved. In v10 API Gateway, `context.get()` always returns an XML document, even if an xml node set was set in the variable. The API may require modification to work properly.
- In both v5 and v10 API Gateway, using the `apim.getvariable()` function on XML data returns an XML NodeList object. In v10 API Gateway, using the `context.get()` function on XML data returns an XML document. Consider this change before switching to `context.get()` using the `optimize-gws` migration configuration option, as API behavior may change as a result.

**Severity**

warn

**GWS05 Added `require('apim')` function to `gatewayscript` policy to conform to the API Gateway specification.**

**Explanation**

See message text.

**Response**

Review the changes to ensure that the `gatewayscript` policy result will be consistent with the v5 API.

**Severity**

info

**GWS06 Found `require('header-metadata');` in the `gatewayscript` source.**

#### Explanation

The `header-metadata` module is not supported in the API Gateway. All API context variables in the API Gateway are in the global context module. You can access the context variables directly using `context.set()` or `context.get()`.  
 For more information about using context variables in `gatewayscript` policies in API Gateway, see [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway](#).

#### Response

For better performance, either use the `--optimize-gws=true` toggle when running the migration utility, or use the context module for native API Gateway performance. For more information about `header-metadata`, see [Access to the headers object](#) in the IBM DataPower Gateways documentation.

#### Severity

info

### GWS07 Replaced `require('local:///isp/policy/apim.custom.js')` with `require('apim')` to conform to the API Gateway specification.

#### Explanation

The v5 `gatewayscript` compatibility module does not exist on the local file system in the API Gateway. The `require` function for this module was present in the `gatewayscript` code and has been changed for API Gateway compatibility.

#### Severity

info

### GWS08 Empty source `'gatewayscript'` found.

#### Explanation

See message text.

#### Response

Consider updating the empty `gatewayscript` policy or remove it from the assembly for better performance.

#### Severity

warn

### GWS09 Found `require('fs')` in the `'gatewayscript'` source.

#### Explanation

The `fs` module is supported in the API Gateway. Some API Gateway configuration folders have restricted access.

#### Severity

info

### GWS10 Found `require('service-metadata')` in the `'gatewayscript'` source.

#### Explanation

See message text.

#### Response

The Gatewayscript examples in the following table show how the service variables map to API Gateway metric functions. To obtain the metric data, use the `getMetricData()` API of the `context.metric` object in DataPower API Gateway, as in the following example.

```
var md = context.metric.getMetricData();
```

For more information about the `context.metric` object, see [context.metric object for API Gateway services](#) in the IBM DataPower Gateways documentation.

| v5 MPGW service variable                                         | Gatewayscript example                                                               | Description                                                |
|------------------------------------------------------------------|-------------------------------------------------------------------------------------|------------------------------------------------------------|
| <code>dp:variable('var://service/time-response-complete')</code> | <code>md.filter(n =&gt; n.type === 'assembly-invoke').slice(-1).endTime</code>      | The end time of the last assembly-invoke metric data.      |
| <code>dp:variable('var://service/time-elapsed')</code>           | <code>md.slice(-1).endTime</code>                                                   | The end time of the last element in the metric data array. |
| <code>dp:variable('var://service/time-forwarded')</code>         | <code>md.filter(n =&gt; n.type === 'assembly-invoke').slice(-1).requestTime</code>  | The request time of the last assembly-invoke metric data.  |
| <code>dp:variable('var://service/URL-out')</code>                | <code>md.filter(n =&gt; n.type === 'assembly-invoke').slice(-1).targetURL</code>    | The target URL of the last assembly-invoke metric data.    |
| <code>dp:variable('var://service/mpgw/request-size')</code>      | <code>md.filter(n =&gt; n.type === 'assembly-invoke').slice(-1).requestSize</code>  | The request size of the last assembly-invoke metric data.  |
| <code>dp:variable('var://service/mpgw/response-size')</code>     | <code>md.filter(n =&gt; n.type === 'assembly-invoke').slice(-1).responseSize</code> | The response size of the last assembly-invoke metric data. |

#### Severity

info

**GWS11 Found `require('mgmt')` in the `gatewayscript` source.**

**Explanation**

The `mgmt` module is not supported in the API Gateway.

**Response**

Remove use of the `mgmt` module from the source code.

**Severity**

error

**GWS12 Found `require('policy/policy-utility')` or `require('policy/policy-utility.js')` in the `gatewayscript` source.**

**Explanation**

`policy-utility` is a DataPower module that is not supported in the API Gateway.

**Response**

Remove use of the `policy-utility` module from the source code.

**Severity**

error

**GWS18 Found `gatewayscript` containing session function: `%s` and replaced with API Gateway native `%s`**

**Explanation**

See message text.

**Severity**

info

**GWS19 Invalid `gatewayscript` found. The gateway script could not be parsed due to `%s`**

**Explanation**

See message text.

**Severity**

warn

**GWS20 Found `gatewayscript` containing apim function: `%s`. No replacement was performed. The closest match is '%s', but it may not work in all situations.**

**Explanation**

See message text.

**Severity**

warn

**GWS21 Found `gatewayscript` containing session context function: `%s`. No replacement was performed. The closest match is '%s', but it may not work in all situations.**

**Explanation**

See message text.

**Severity**

warn

**GWS22 Found `gatewayscript` containing: `%s` and replaced with API Gateway equivalent context var of: `%s`**

**Explanation**

The following context variable names differ between v5 and v10 API Gateway. The migration utility replaces the v5 variable name with the equivalent v10 API Gateway variable name. For more information, see [API Connect context variables](#).

| v5 context variable   | v10 API Gateway context variable |
|-----------------------|----------------------------------|
| <code>env.path</code> | <code>api.catalog.path</code>    |

| v5 context variable          | v10 API Gateway context variable        |
|------------------------------|-----------------------------------------|
| <code>env.name</code>        | <code>api.catalog.name</code>           |
| <code>client.app.type</code> | <code>client.app.lifecycle-state</code> |

**Severity**

info

**GWS24 Found gatewayscript policy containing api.document. Adding a gatewayscript policy to populate the api.document context.**

**Explanation**

The migration utility added a gatewayscript policy before the gatewayscript policy that contains `api.document`. The additional gatewayscript policy is required to populate the `api.document` context.

**Severity**

info

**GWS25 Replaced TLS Profile name `webapi-sslcli-mgmt` or `api-sslcli-all` with default TLS profile name `api-ssl-client-all` to conform to the API Gateway specification.**

**Explanation**

See message text.

**Severity**

info

**GWS26 Removed `%s \\\` characters to conform to the API Gateway specification.**

**Explanation**

The processing of v5 API metadata would implicitly remove extraneous `\\` characters. The source property of the gatewayscript policy has been changed to removed these characters as they would have been removed in v5.

**Severity**

info

**Invoke policy messages**

- [INV01](#)
- [INV02](#)
- [INV03](#)
- [INV04](#)
- [INV05](#)
- [INV06](#)
- [INV07](#)
- [INV08](#)
- [INV09](#)
- [INV10](#)
- [INV11](#)
- [INV12](#)
- [INV13](#)
- [INV14](#)
- [INV15](#)
- [INV16](#)
- [INV17](#)
- [INV18](#)
- 

**INV01 Moved `x-ibm-gateway-decode-request-params` from API properties to each `invoke` policy.**

**Explanation**

This v5 API property only applies to the `invoke` and `proxy` policies. In API Gateway, the functionality of this property is implemented as an option in the `invoke` policy. This change was made for the following purposes.

- The option is now clearly visible in the `invoke` policy, making it easy to identify available options.
- Each use of the `invoke` policy can now have a different setting.

**Severity**

info

**INV02 Moved `x-ibm-gateway-invoke-keep-payload` from API properties to the `keep-payload` `invoke` property where applicable.**

**Explanation**



This v5 API property applies only to the `invoke` policy. In API Gateway, the functionality of this property is implemented as an option in the `invoke` policy. This change was made for the following purposes:

- The option is now clearly visible in the `invoke` policy, making it easy to identify available options.
- Each use of the `invoke` policy can now have a different setting.

**Severity**

warn

**INV03 Moved `x-ibm-gateway-queryparam-encode-plus-char` from API properties to each `invoke` policy.**

**Explanation**

This API property only applies to the `Invoke` policy. In API gateway, this property is an option in the `Invoke` policy. This change was made for the following purposes:

- The option is now clearly visible on the `invoke` policy, making it easy to identify available options.
- Each use of the `invoke` policy can now have a different setting.

**Severity**

info

**INV04 Added `parse` policy immediately after the `invoke` policy to parse the `invoke` output.**

**Explanation**

The `invoke` policy output context in the API Gateway is not parsed. For compatibility with the v5 API, the `invoke` policy output context will be explicitly parsed by a `parse` policy that has been added to execute immediately after the `invoke` policy.

**Severity**

info

**INV05 Removed `x-ibm-gateway-invoke-suppress-clientid` API property from the YAML file and replaced it with allowlists and blocklists to conform to the API Gateway specification.**

**Explanation**

This v5 API property applies only to the `invoke` policy. In API Gateway, the functionality of this property is implemented as an option in the `invoke` policy. This change was made for the following purposes:

- The option is now clearly visible in the `invoke` policy, making it easy to identify available options.
- Each use of the `invoke` policy can now have a different setting.

**Severity**

warn

**INV06 Found `invoke` policy. Converted to API Gateway `invoke` policy.**

**Explanation**

See message text.

**Severity**

info

**INV07 The API property `x-ibm-gateway-invoke-emulate-v4-soap-error` does not exist in the API Gateway.**

**Explanation**

A `x-ibm-gateway-invoke-emulate-v4-soap-error` property was found. This property does not exist in the API gateway.

**Response**

Remove the `x-ibm-gateway-invoke-emulate-v4-soap-error` property from the API.

**Severity**

error

**INV08 Incorrect timeout value of `%s` found in `%s` policy, updated to `60` seconds.**

**Explanation**

The `timeout` property of the v5 `invoke` policy is invalid. It must be a numeric data type. It has been set to the default value of `60`.

**Response**

Correct the timeout value if the default value does not satisfy the requirements of the policy.

**Severity**

info

**INV09 Renamed the `keepPayload` property in the `invoke` policy to `keep-payload`.**

**Explanation**

See message text.

**Severity**

info

**INV10 `X-IBM-Client-Id` HTTP header suppressed using header-control blocklist, as the `x-ibm-gateway-invoke-suppress-clientid` property is undefined or not set in the v5-compatible YAML input file.**

**Explanation**

The v5 default behavior was to not send the `X-IBM-Client-Id` HTTP header to the invoke target. The `x-ibm-gateway-invoke-suppress-clientid` is not provided, so the header will not be sent. The API Gateway accomplishes this behavior by adding the header to the `invoke` policy header-control blocklist.

**Severity**

info

**INV11 `X-IBM-Client-Id` HTTP header suppressed using header-control blocklist, as the `x-ibm-gateway-invoke-suppress-clientid` property is set to `true` in the v5-compatible YAML input file.**

**Explanation**

The v5 default behavior was to not send the `X-IBM-Client-Id` HTTP header to the invoke target. The `x-ibm-gateway-invoke-suppress-clientid` property is provided with a value of `true`, so the header will not be sent. The API Gateway accomplishes this behavior by adding the header to the `invoke` policy header-control blocklist.

**Severity**

info

**INV12 Removed `x-ibm-gateway-optimize-invoke` API property from the YAML file and converted the `invoke` policy to conform to the API Gateway specification.**

**Explanation**

This v5 API property is not supported by the API Gateway.

**Severity**

info

**INV13 Added `set-variable` policy to preserve `X-IBM-Client-Id` HTTP header before the `invoke` policy, as `x-ibm-gateway-invoke-suppress-clientid` is set to `false` in the v5c YAML input file.**

**Explanation**

By default in the API Gateway, the `X-IBM-Client-Id` HTTP header is not copied from the inbound request headers to the message context headers used by the `invoke` policy. This request header will be copied to the message context headers so that it will be available to be sent by the `invoke` policy.

**Severity**

warn

**INV14 Added `gatewayscript` policy for `x-ibm-gateway-invoke-v4-nomap-emulation` as the property is set to `true` in the v5-compatible YAML input file.**

**Explanation**

The v5 `invoke` policy used this property internally, but this behavior is not supported by the API Gateway `invoke` policy. A `gatewayscript` policy that implements the behavior of this v5 API property has been added to execute after the `invoke` policy.

**Severity**

info

**INV15 Invoke `keep-payload` property has been set to the value of the API property `x-ibm-gateway-invoke-keep-payload`.**

**Explanation**

Although not supported by the v5 UI, a `keep-payload` property in the `invoke` policy is always used if present to control this behavior. If not present in the policy, the API property is used instead. The policy property is not present, but the API property is present, therefore the `keep-payload` property in the API Gateway will use the v5 API property.

**Severity**

info

**INV16 Set `keep-payload` setting to `true`.**

**Explanation**

The `keep-payload` setting is set based on the following conditions:

- For an `invoke` policy that meets all of the following criteria, the `keep-payload` property is always set to `true`.
  - The policy is at the end of the assembly.
  - The API property `x-ibm-gateway-optimize-invoke` is not set to `false`.
  - The `invoke` policy does not contain a `stop-on-error` property.
- For an `invoke` policy that does not meet the above criteria, the `keep-payload property` is set to one of the following values, in order of precedence.
  - The value of `keepPayload` property, if it is set.
  - The value of the API property `x-ibm-gateway-invoke-keep-payload`, if it is set.
- For an `invoke` policy that meets none of the above criteria, the `keep-payload` property is not set at all.
- For a `proxy` policy, the `keep-payload` property is always set to `true` after it is converted to an `invoke` policy.

**Severity**

info

**INV17 Invoke policy set to `invoke\_1.5.0`, converting invoke to `invoke\_1.5.0`**

**Explanation**

See message text.

**Severity**

info

**INV18 Set `final-action` parameter on `invoke\_1.5.0`**

**Explanation**

If the `invoke_1.5.0` policy is last in the assembly, the migration utility adds the `final-action` parameter, which ensures that the policy sends output to the message context. If `invoke_1.5.0` is not last in the assembly, the policy adds a `parse` after the wrapper policy to parse the output.

**Severity**

info

**JSON-to-XML policy messages**

- [J2X01](#)
- [J2X02](#)

**J2X01 Added a `parse` policy to execute before the `json-to-xml` policy.**

**Explanation**

A v5 `json-to-xml` policy must have the current API Gateway context parsed explicitly prior to execution. The `parse` policy has been added to execute before the `json-to-xml` policy.

**Severity**

info

**J2X02 Found `json-to-xml` policy and converted to API Gateway `json-to-xml`.**

**Explanation**

See message text.

**Severity**

info

**JWT-Generate policy messages**

- [JTG01](#)
- [JTG02](#)
- [JTG03](#)

### JTG01 JWE properties are missing in the `jwt-generate` policy, so encryption has been disabled.

#### Explanation

If `jwt-enc` or `jwt-alg` is specified, then at least one of `jwt-jwk` and `jwt-crypto` must also be specified.

#### Severity

warn

### JTG02 JWS property is missing in the `jwt-generate` policy, so signing has been disabled.

#### Explanation

If `jwt-alg` is specified, at least one of `jwt-jwk` or `jwt-crypto` must also be specified.  
A private key must be specified, which can be either a JSON web key or a crypto object.

#### Severity

warn

### JTG03 Converted the `jwt-generate` policy to the API Gateway `jwt-generate` policy.

#### Explanation

See message text.

#### Severity

info

### LTPA-Generate policy messages

- [LTPA01](#)

### LTPA01 Found v5-compatible `ltpa-generate` policy. Because this policy does not exist in the API Gateway, it is unchanged.

#### Explanation

The `ltpa-generate` policy is not available in the API Gateway.

#### Response

Remove the `ltpa-generate` policy from the API.

#### Severity

error

### Map policy messages

- [MAP01](#)
- [MAP02](#)
- [MAP03](#)
- [MAP04](#)
- [MAP05](#)
- [MAP06](#)

### MAP01 "Moved `%s` from API properties to each `map` policy as map option: `%s`."

#### Explanation

This v5 API property applies only to the `map` policy. In API Gateway, the functionality of this property is implemented as an option in the `map` policy. This change was made for the following purposes.

- The option is now clearly visible on the `map` policy, making it easy to identify available options.
- Each use of the `map` policy can now have a different setting.

#### Severity

info

### MAP02 Replaced %s with %s to conform to the API Gateway syntax specification.

#### Explanation

API Connect Version 5 treats a map value property with a string quoted with `\` or `\\` as the same, specifically as a double quote. This is not supported in the API Gateway. The `\\` is converted to a single `\`.

#### Severity

info

### MAP03 Converted `map` to API Gateway `map` for conformance."

#### Explanation

See message text.

#### Severity

info

### MAP04 Set `map` option of `mapResolveApicVariables` to previous default of `true` for compatibility.

#### Explanation

If a map policy is found to not have `mapResolveApicVariables` set, the option is set to true to maintain backwards compatibility.

#### Severity

info

### MAP05 Set `map` option of `mapParseXMLOutput` to `true` for compatibility.

#### Explanation

This property is always set by the migration utility to maintain compatibility with v5. If XML data is output from the `map\_2.0.0` policy, then the data is parsed before it reaches `<output>.body`.

#### Severity

info

### MAP06 Removed %s `\\` characters to conform to the API Gateway specification.

#### Explanation

The processing of v5 API metadata would implicitly remove extraneous `\\` characters. The value properties of the map policy action have been changed to remove these characters as they would have been removed in v5.

#### Severity

info

### Proxy policy messages

- [PRX01](#)
- [PRX02](#)
- [PRX03](#)
- [PRX04](#)
- [PRX05](#)
- [PRX06](#)
- [PRX07](#)
- [PRX08](#)
- [PRX09](#)
- [PRX10](#)
- [PRX11](#)
- [PRX12](#)
- [PRX13](#)
- [PRX14](#)
- [PRX15](#)

### PRX01 Found `proxy` policy and converted to API Gateway `invoke` policy, as `proxy` does not exist in the API Gateway.

#### Explanation

A `proxy` policy was found. The policy was converted to an `invoke` policy in the API gateway.

#### Response

Review the changes to ensure that the `invoke` policy result will be consistent with the v5 API. Alternatively, you can specify that the migration utility convert `proxy` policies to `proxy_1.5.0`, which is a backward-compatible policy available in the API Gateway.

#### Severity

info

### PRX02 Removed `x-ibm-gateway-proxy-suppress-clientid` API property from the YAML file and replaced it with allowlists and blocklists to conform to the API Gateway specification.

**Explanation**

This v5 API property applies only to the `proxy` policy. In the API Gateway, this property is represented by the header control and parameter control options in the `invoke` policy to either allow or restrict the `X-IBM-Client-Id` header or `client_id` parameter being sent to the backend server. This change was made for the following reasons.

- The option is now clearly visible in the `invoke` policy, making it easy to identify available options.
- Each use of the `invoke` policy can now have a different setting.

**Severity**

warn

**PRX03 Set policy `title` to `invoke` as this is now an `invoke` policy.****Explanation**

The functionality of the v5 `invoke` and `proxy` policies has been merged into the API Gateway `invoke` policy. The `title` property of the policy was changed to `invoke`.

**Severity**

info

**PRX04 Proxy: Changed `No-Cache` to lowercase `no-cache`.****Explanation**

See message text.

**Severity**

info

**PRX05 Proxy: Changed `Time-to-Live` to lowercase `time-to-live`.****Explanation**

See message text.

**Severity**

info

**PRX06 Proxy: Changed `Protocol-Based` to lowercase `protocol`.****Explanation**

See message text.

**Severity**

info

**PRX07 `client\_id` query parameter suppressed using parameter-control blacklist, as `x-ibm-gateway-proxy-suppress-clientid` is set to `true` in the v5-compatible YAML input file.****Explanation**

The v5 default behavior is to send the `client_id` query parameter to the proxy target. The `x-ibm-gateway-proxy-suppress-clientid` property was provided with a value of `true`, indicating not to send this query parameter. The API Gateway accomplishes this behavior by adding this query parameter name to the parameter-control blacklist of the policy.

**Severity**

info

**PRX08 `X-IBM-Client-Id` HTTP header suppressed using header-control blacklist, as `x-ibm-gateway-proxy-suppress-clientid` is set to `true` in the v5-compatible YAML input file.****Explanation**

The v5 default behavior was to not send the `X-IBM-Client-Id` HTTP header to the proxy target. The `x-ibm-gateway-proxy-suppress-clientid` property was provided with a value of `true`, indicating not to send this HTTP header. The API Gateway accomplishes this behavior by adding this header name to the header-control blacklist of the policy.

**Severity**

info

**PRX09 `X-IBM-Client-Id` HTTP header suppressed using header-control blacklist, as `x-ibm-gateway-proxy-suppress-clientid` is undefined or not set in the v5-compatible YAML input file.**

**Explanation**

The v5 default behavior was to not send the `X-IBM-Client-Id` HTTP header to the proxy target. The `x-ibm-gateway-proxy-suppress-clientid` property was not provided, indicating not to send this HTTP header. The API Gateway accomplishes this behavior by adding this header name to the header-control blacklist of the policy.

**Severity**

info

**PRX10 `client\_id` not suppressed in query parameter, as `x-ibm-gateway-proxy-suppress-clientid` is undefined or not set in the v5-compatible YAML input file.**

**Explanation**

The v5 default behavior was to send the `client_id` query parameter to the proxy target. The `x-ibm-gateway-proxy-suppress-clientid` property was not provided, indicating to send this query parameter. No API Gateway `invoke` policy configuration is required to emulate this behavior.

**Severity**

warn

**PRX11 `output` deleted as this policy is the last policy in the assembly flow.**

**Explanation**

The v5 `proxy` API can specify an output context that makes the `proxy` response available for subsequent policies, but that response was also made available as the API response, should a subsequent policy not update the message context. When the `proxy` policy is the last policy in the v5 assembly, this output is removed, which allows the API Gateway `invoke` response to be returned to the client.

**Severity**

info

**PRX12 Added `gatewayscript` policy to copy the proxy policy headers, status, and output to `message` context.**

**Explanation**

A v5 `proxy` policy that specifies an output response context name is not the last policy in the assembly flow. A `gatewayscript` policy has been added after the API Gateway `invoke` policy. The `gatewayscript` policy copies the contents of the response context to the message context to emulate v5 behavior.

**Severity**

warn

**PRX13 Added `set-variable` policy to preserve `X-IBM-Client-Id` HTTP header immediately before the proxy as `x-ibm-gateway-proxy-suppress-clientid` was set to `false` in the V5-compatible YAML input file.**

**Explanation**

By default in the API Gateway, the `X-IBM-Client-Id` HTTP header is not copied from the inbound request headers to the message context headers used by the `invoke` policy. This request header will be copied to the message context headers so that it will be available to be sent by the `invoke` policy.

**Severity**

info

**PRX14 Proxy policy set to `proxy\_1.5.0`, converting proxy to `proxy\_1.5.0`**

**Explanation**

See message text.

**Severity**

info

**PRX15 Set `final-action` parameter on `proxy\_1.5.0`**

**Explanation**

If the `proxy_1.5.0` policy is last in the assembly, the migration utility adds the `final-action` parameter, which ensures that the policy sends output to the message context. If `proxy_1.5.0` is not last in the assembly, the policy adds a parse after the wrapper policy to parse the output.

**Severity**

info

## Redact policy messages

- [RE01](#)
- [RE02](#)
- [RE03](#)
- [RE04](#)
- [RE05](#)
- [RE06](#)
- [RE07](#)
- [RE08](#)
- [RE09](#)
- [RE10](#)
- [RE11](#)
- [RE12](#)
- [RE13](#)
- [RE14](#)

### **RE01 This API Gateway `redact` policy is added to contain all v5-compatible `redact` paths targeted against the `request`.**

#### Explanation

To redact the request body, an API Gateway `redact` policy is placed at the front of the assembly in order to redact incoming data. All redactions in v5c that are not found within a `switch`, `if`, or `operation-switch` component will be moved to this single `redact` policy.

#### Response

Review to confirm that this change meets the purpose of the API.

#### Severity

warn

### **RE02 API Gateway `redact` policy added to contain all v5-compatible `redact` paths targeted against `response`, `logs`, and `all`.**

#### Explanation

In the API Gateway, `response` and `logs` data must be redacted at the end of the assembly. This API Gateway `redact` policy was added to the end of the assembly to conform to the specification. If `logs` or `all` is the v5-compatible target for the `redact` policy, then a new `log` policy will be added in `gather-only` mode to populate the `log` API context.

The API Gateway emulates the v5-compatible redact on response and logs by adding a redact action at the end of the assembly to redact the response. All redactions in v5c that are not found inside of a `switch`, `if`, or `operation-switch` component will be moved to this single `redact` policy.

#### Response

Review to confirm that this change meets the purpose of the API.

#### Severity

warn

### **RE03 Found v5-compatible `redact` policy. `action:path` pairs from this policy will be sorted into one or both leading and trailing API Gateway `redact` policies to conform to the API Gateway specification.**

#### Explanation

To emulate the v5 redact on `request` and `response`, some of the `redact` policies must be moved to either the end of the assembly or beginning of the assembly. This change is necessary because the `redact` policy now works directly on the API context, so it must be placed in a specific order so that it can redact data either before it enters the assembly or after it exits the assembly.

#### Response

Review the changes to ensure that the `redact` policy result will be consistent with the v5 API.

#### Severity

warn

### **RE04 Found duplicate v5-compatible `redact` path of the same `redact` type. Duplicate item will not be added.**

#### Explanation

Because multiple redactions are consolidated into the leading or trailing `redact` policy, there might be some paths that are duplicates of current paths in the consolidated `redact` policy. These paths are not added as they are redundant.

#### Response

Review the changes to ensure that the `redact` policy result will be consistent with the v5 API.

#### Severity



warn

**RE05 Redaction on `request` was found in the v5-compatible YAML input file. Added first API Gateway `redact` policy to the front of the assembly after the `parse` policy to perform this redaction.**

**Explanation**

To redact the request body, an API Gateway `redact` policy is placed at the front of the assembly in order to redact incoming data. All redactions in v5c that are not found within a `switch`, `if`, or `operation-switch` component will be moved to this single `redact` policy.

**Response**

Review to confirm that this change meets the purpose of the API.

**Severity**

info

**RE06 Redact on `response`, `logs`, or `all` was found in the v5-compatible YAML input file. Added second API Gateway `redact` to the end of the assembly to perform this redaction. `log` policy also added if redact on `logs` or `all` was found.**

**Explanation**

In the API Gateway, `response` and `logs` data must be redacted at the end of the assembly. This API Gateway `redact` policy was added to the end of the assembly to conform to the specification. If `logs` or `all` is the v5-compatible target for the `redact` policy, then a new `log` policy will be added in `gather-only` mode to populate the `log` API context.

The API Gateway emulates the v5-compatible redact on response and logs by adding a redact action at the end of the assembly to redact the response. All redactions in v5c that are not found inside of a `switch`, `if`, or `operation-switch` component will be moved to this single `redact` policy.

**Response**

Review to confirm that this change meets the purpose of the API.

**Severity**

warn

**RE07 Potential JSON v5-compatible `redact` path detected.**

**Explanation**

A v5 `redact` path has been detected that is highly likely to be targeted against JSON data. It has been converted to a JSONata path because the `redact` policy in the API Gateway uses JSONata to specify `path`.

**Response**

Review to confirm that the path is meant for JSON data and it is formed properly.

**Severity**

warn

**RE08 XML v5-compatible `redact` path detected. Converted to `redact` XPath: `%s` to JSONata: `%s`.**

**Explanation**

A v5-compatible `redact` path has been detected that is highly likely to be targeted against XML data. It has been converted to JSONata to meet the API Gateway specification. The `redact` policy in the API Gateway uses JSONata for `path`.

**Response**

Review to confirm that the path is meant for XML data and it is formed properly. Note that a malformed XPath may still publish on API Connect v10 API Gateway, but will fail subsequent API calls with a **404 API Not Found** error. The API Definition DataPower object will also be in the `[down]` state because of the invalid XPath. It is also possible for an invalid XPath to work on v5 but fail on API Gateway. For example, the XPath `/test/path/` is invalid because of the trailing slash. However, this XPath will work in v5, and fail with the above scenario in v10 API Gateway.

**Severity**

warn

**RE09 Common `redact` XPath `%s` replaced with JSONata `%s` to adhere to the API Gateway `redact` specification.**

**Explanation**

The `redact` policy in API Gateway uses JSONata to specify the `path`. All v5-compatible `redact` paths are converted to JSONata.

**Response**

Review to confirm that the changes conform to the purpose of the `redact` policy.

**Severity**

info

**RE10 Invalid redact action: `%s` found.****Explanation**

A v5-compatible `redact` policy was found to contain an invalid `action` type. Valid types include `redact` or `remove`.

**Response**

Review the `action` type in the `redact` policy and update to either `redact` or `remove`.

**Severity**

error

**RE11 Added `log` policy to populate the `log` API context, because a v5-compatible `redact` policy was found targeted against `logs` or `all`. An API Gateway `redact` policy will be appended after this policy to redact the outgoing `log` data.****Explanation**

The API Gateway specification requires a `log` policy in the assembly to populate the `log.request_body` and `log.response_body` context variables. After the `log` policy, the `redact` policy will then redact the contents of `log.request_body` and `log.response_body`.

**Severity**

warn

**RE12 Added `parse` policy before `redact` policy to parse incoming data.****Explanation**

The incoming data must be parsed for the `redact` policy to work properly. For example, if the incoming data contains unparsed but well-formed XML data, the `redact` policy does nothing and the data passes through. With this added `parse` policy, the XML data is parsed properly and the redaction works as expected.

**Response**

Review the changes to ensure that the `redact` policy result will be consistent with the v5 API.

**Severity**

warn

**RE13 Found v5-compatible `redact` policy inside of a `switch` policy. Converted all paths to JSONata.****Explanation**

If a v5-compatible `redact` policy is found within a `switch`, `if`, or `operation-switch` component, it will be left in place. If the redaction is on the response and there is an `invoke` policy at the end of the assembly, problems could arise.

**Response**

Review and confirm that this redaction conforms to the purpose of the API.

**Severity**

warn

**RE14 Invalid `redact` action found: path is empty.****Explanation**

The v5-compatible `redact` policy was found to have an empty path. This configuration is invalid

**Response**

To make the API functional, update the path in the `redact` policy.

**Severity**

error

**Set-Variable policy messages**

- [SV01](#)
- [SV02](#)
- [SV05](#)
- [SV06](#)

- [SV07](#)
- [SV08](#)

**SV01 Found `set-variable` policy and converted to API Gateway `set-variable` policy.**

**Explanation**

See message text.

**Severity**

info

**SV02 Cannot recognize `set-variable` action ` %s`. It must be `set`, `add`, or `clear` with a valid variable name.**

**Explanation**

The v5 `set-variable` policy contains an action that is not `set`, `add`, or `clear`. The API Gateway `set-variable` policy should be modified to use the correct settings.

**Severity**

error

**SV05 Required property `type` not found. Added `type` of `any` to conform to the API Gateway specification.**

**Explanation**

The v5 `set-variable` policy does not have a `type` property. The type is inherent in the data type of the value. The API Gateway includes this property, which is set to the data type of the value. In conditions where the data type would be determined at runtime (for example, with a variable substitution), the value is set to `any`.

**Response**

Consider changing the added `type` to a more specific `type`.

**Severity**

warn

**SV06 Required property `type` not found. Added `type` of ` %s` to conform to the API Gateway specification, and it is consistent with the provided `value`.**

**Explanation**

See message text.

**Severity**

info

**SV07 The set-variable policy has no valid actions and has been removed from the assembly.**

**Explanation**

See message text.

**Severity**

info

**SV08 The set-variable action name '%s' has spaces in the name. Changing the action name to '%s'.**

**Explanation**

See message text.

**Severity**

info

**Switch policy messages**

- [SW01](#)
- [SW03](#)
- [SW04](#)
- [SW05](#)
- [SW06](#)
- [SW07](#)
- [SW08](#)
- [SW09](#)

- [SW10](#)
- [SW11](#)
- [SW12](#)
- [SW13](#)

**SW01 Found `operation-switch\_1.0.0` and converted to API Gateway `operation-switch\_2.0.0`.**

**Explanation**

See message text.

**Severity**

info

**SW03 Found `switch` policy and converted to the API Gateway `switch` policy.**

**Explanation**

A `switch` policy from a v5 API is converted to an API Gateway `switch` policy. The v5 API uses GatewayScript syntax for conditions, whereas the API Gateway uses JSONata syntax. The conditions have been moved to a `gatewayscript` policy that will execute immediately before the `switch` policy. The `switch` conditions will use results from the `gatewayscript` policy to execute the correct assembly path.

**Response**

Review the changes to ensure that the `switch` policy result will be consistent with the v5 API.

**Severity**

info

**SW04 Found `if` policy and converted to API Gateway `switch` policy.**

**Explanation**

An `if` policy from a v5 API is converted to an API Gateway `switch` policy. The v5 API uses GatewayScript syntax for conditions, whereas the API Gateway uses JSONata syntax. The conditions have been moved to a `gatewayscript` policy that will execute immediately before the `switch` policy. The `switch` conditions will use results from the `gatewayscript` policy to execute the correct assembly path.

**Severity**

info

**SW05 Found `switch` condition: `%s` and converted it to JSONata: `%s` to conform to the API Gateway specification.**

**Explanation**

A v5 `switch` policy condition was converted directly into a JSONata condition. This condition does not require a `gatewayscript` policy that would be executed prior to the switch.

**Severity**

warn

**SW06 `switch` condition: `%s` could not be converted to JSONata. (Error code: `%d`).**

**Explanation**

Cannot convert an `if/switch` condition to JSONata. A GatewayScript policy will be added to handle the Switch logic instead. Changes are needed only if highest performance is desired. See the Response section for more details on performance. This message can appear for the following reasons.

| Error code | Explanation                                                             |
|------------|-------------------------------------------------------------------------|
| 1          | The auto-generated JSONata mapping file was not found.                  |
| 2          | The condition was not found in the auto-generated JSONata mapping file. |
| 3          | The condition resolved to empty.                                        |
| 4          | An unknown error occurred                                               |

**Response**

To improve performance, review the condition in the v5 API and provide the JSONata for this condition.

If an `indexOf()` method is used to determine whether a substring is present in a value, consider using the `$contains()` JSONata string function. If `indexOf()` is used to get a substring before or after the match, consider using the `substringBefore()` or `substringAfter()` JSONata string functions.

For more information about using JSONata to write switch conditions, see [Writing switch condition scripts - DataPower API Gateway](#).

**Severity**

warn

**SW07 `switch` condition: `%s` could not be converted to JSONata or handled by `gatewayscript`.**

**Explanation**

The `if/switch` condition cannot be converted to JSONata. This error can occur for the following reasons:

- A condition contains invalid Javascript.
- The condition contains multiple expressions.

**Response**

Review the contents of the condition and edit as necessary.

**Severity**

error

**SW08 `switch` policy set to `switch\_1.5.0`, converting `switch` to `switch\_1.5.0`.**

**Explanation**

See message text.

**Severity**

info

**SW09 `if` policy set to `if\_1.5.0`, converting `if` to `if\_1.5.0`.**

**Explanation**

See message text.

**Severity**

info

**SW10 Replaced `\\\" with `\"` %d times to conform to the JavaScript syntax specification.**

**Explanation**

See message text.

**Severity**

info

**SW11 Condition: %s could not be parsed %s.**

**Explanation**

See message text.

**Severity**

error

**SW12 Condition: %s contains inline parameters %s within a string. These references cannot be resolved.**

**Explanation**

See message text.

**Severity**

error

**SW13 Empty switch condition found and will be ignored.**

**Explanation**

See message text.

**Severity**

info

**Throw policy messages**

- [TH01](#)

**TH01 Found `throw` policy and converted to API Gateway `throw` policy to conform to the API Gateway specification.**

**Explanation**

See message text.

**Severity**

info

**Validate policy messages**

- [VAL01](#)
- [VAL02](#)
- [VAL03](#)
- [VAL04](#)
- [VAL05](#)
- [VAL06](#)
- [VAL07](#)
- [VAL08](#)

**VAL01 Found v5-compatible `validate-usenametoken` policy. This policy does not exist in the API Gateway. No action has been taken.**

**Explanation**

A `validate-usenametoken` policy was found. This policy does not exist in the api gateway and has been ignored.

**Response**

Remove the `validate-usenametoken` policy from the API.

**Severity**

error

**VAL02 Converted `validate` policy into an API Gateway `validate` policy.**

**Explanation**

A `validate` policy was found.

**Response**

Review the changes to ensure that the `validate` policy result will be consistent with the v5 API.

**Severity**

info

**VAL03 Found `definition: request` and replaced it with `validate-against: body-param`.**

**Explanation**

The `definition` property in the v5 `validate` policy contained a value of `request`. This value has been changed to a value of `body-param` in the `validate-against` property of the API Gateway `validate` policy.

**Severity**

info

**VAL04 Found `definition: response` and replaced it with `validate-against: response-param`.**

**Explanation**

The `definition` property of the v5 `validate` policy contained a value of `request`. This value has been changed to a value of `response-param` in the `validate-against` property of the API Gateway `validate` policy.

**Severity**

info

**VAL05 Found `soap-env` `validate` definition and converted it to the API Gateway `xml-schema` property.**

**Explanation**

The `definition` property in the v5 `validate` policy references an OpenAPI definition. The definition referenced is for a SOAP document. This value has been changed to a value of `url` in the `validate-against` property in the API Gateway `validate` policy. The `definition` property value has been moved to the `xml-schema` property.

**Severity**

info

### VAL06 Set `xml-validation-mode` to `soap-body`.

#### Explanation

The `definition` property in the v5 `validate` policy references an OpenAPI definition. The definition referenced is for a SOAP document. This value has been changed to a value of `url` in the `validate-against` property in the API Gateway `validate` policy. The `xml-validation-mode` property value has been set to `soap-body`.

#### Severity

info

### VAL07 Set `validate-against` to `%s`.

#### Explanation

The `definition` property in the v5 `validate` policy references an OpenAPI definition. This value has been changed to conform to the API Gateway `validate` policy property `validate-against` with a value of `definition` or `url`.

#### Severity

info

### VAL08 Added `parse` policy before `validate` policy.

#### Explanation

A v5 `validate` policy that is configured to validate the current context payload against a schema definition must have the input of the current API Gateway context parsed explicitly prior to execution. The `parse` policy has been added to execute before the `validate` policy.

#### Severity

info

### WSDL messages

- [WSDL01](#)
- [WSDL02](#)
- [WSDL03](#)
- [WSDL04](#)
- [WSDL05](#)
- [WSDL06](#)
- [WSDL07](#)
- [WSDL08](#)
- [WSDL09](#)

### WSDL01 An API created from WSDL does not have the required field `%s`.

#### Explanation

See message text.

#### Severity

error

### WSDL02 The WSDL file `%s` does not exist. This file is referenced by field `%s`.

#### Explanation

See message text.

#### Severity

error

### WSDL03 An API created from WSDL has excessive examples. The total length of the examples is %d. This may cause a problem during publish.

#### Explanation

See message text.

#### Severity

warn

### WSDL04 An API created from WSDL has excessive examples. The total length of the examples is %d. The examples are removed.

#### Explanation

See message text.

**Severity**

warn

**WSDL05 A severe error occurred while analyzing an API created from WSDL. [%s]**

**Explanation**

See message text.

**Severity**

error

**WSDL06 An error occurred while analyzing an API created from WSDL. [%s]**

**Explanation**

See message text.

**Severity**

warn

**WSDL07 An API created from WSDL has an embedded error, which was issued when the API was originally created. The error may be fixed in the current version of API Connect. You may want to recreate this API from WSDL. The embedded error [%s] is located at `#/x-ibm-configuration/x-ibm-apiconnect-wsdl/messages/error`.**

**Explanation**

See message text.

**Severity**

warn

**WSDL08 An API created from WSDL is updated to use a new zip (%s). The original zip is located at (%s). Unnecessary files have been removed from the new zip.**

**Explanation**

See message text.

**Severity**

info

**WSDL09 An API was created from a WSDL (%s). Analysis indicates that this zip file may contain unnecessary 'xsd' and 'wsdl' files. Removing the unnecessary files could reduce the size of the zip file by %d bytes.**

**Explanation**

See message text.

**Severity**

info

**XML-to-JSON policy messages**

- [X2J01](#)
- [X2J02](#)

**X2J01 Added a `parse` policy to execute before the `xml-to-json` policy.**

**Explanation**

A v5 `xml-to-json` policy must have the current API Gateway context parsed explicitly prior to execution. The `parse` policy has been added to execute before the `xml-to-json` policy.

**Severity**

info

**X2J02 Found `xml-to-json` policy and converted to API Gateway `xml-to-json` policy.**

**Explanation**



v5 `xml-to-json` policies are converted to API Gateway `xml-to-json` policies.

Note: When a v5 `xml-to-json` policy that contains the `xml:space="preserve"` attribute is migrated to API Gateway, spaces are not retained after curly braces, commas, and quotes in the JSON.

#### Response

Review the JSON and edit as necessary.

#### Severity

info

### XSLT policy messages

- [XSLT01](#)
- [XSLT02](#)
- [XSLT03](#)
- [XSLT04](#)
- [XSLT05](#)
- [XSLT06](#)

#### XSLT01 Replaced ``apim.custom``, ``apim.setvariable``, and ``webapi-util`` with ``apim.custom.xsl`` within `xsl import` and `xsl include` statements.

##### Explanation

The v5 `xslt` policy code that contains an `xsl import` or `xsl include` of v5 framework stylesheets that were found in the local directory of the file system have been changed to use the v5-compatible stylesheet in the store directory of the file system.

##### Severity

info

#### XSLT02 Added ``parse`` policy to execute before the ``xslt`` policy.

##### Explanation

A v5 `xslt` policy that is configured to use the current context payload as input to the stylesheet must have the current API Gateway context parsed explicitly prior to execution. The `parse` policy has been added to execute before the `xslt` policy.

##### Severity

info

#### XSLT03 Found ``xslt`` policy. Porting might be required.

##### Explanation

Review the `xslt` policy and make any changes required for use in API Gateway.

##### Response

If you want to use v5-compatible XSLT code in API Gateway, use the API Gateway extension elements and functions to ensure compatibility with API Gateway. Depending on the content of your code, you may need to make the following changes.

- Remove the `apim.setvariable-impl.xsl` stylesheet from the XSLT code. This stylesheet is not used in API Gateway. Instead, use the API Gateway extension functions to manipulate the assembly context.
- Use API Gateway extension elements to manipulate values in the assembly context. The following example uses the `apigw:set-header` extension element to set the value of the request header.

```
<apigw:set-header name="'request'" value="$request" />
```

- Use of `dp:variable` and `dp:set-variable` functions is not supported in API Gateway. These functions should be changed to `apim:getVariable` and `apim:setVariable`. Access to session, service, and system metadata is restricted.

For more information about the API Gateway extension elements and functions, see the IBM DataPower Gateway documentation.

##### Severity

warn

#### XSLT04 ``xslt`` policy set to ``xslt_1.0.0``, converting `xslt` to ``xslt_1.0.0``.

##### Explanation

See message text.

##### Severity

info

**XSLT05 Replaced TLS Profile name `webapi-sslcli-mgmt` or `api-sslcli-all` with default TLS profile name `api-ssl-client-all` to conform to the API Gateway specification.**

**Explanation**

See message text.

**Severity**

info

**XSLT06 Removed %s `\\` characters to conform to the API Gateway specification.**

**Explanation**

The processing of v5 API metadata would implicitly remove extraneous \\ characters. The source property of the xslt policy has been changed to remove these characters as they would have been removed in v5.

**Severity**

info

## Migration utility messages

---

- [Catalog property updated with catalog name](#)

**Catalog property updated with catalog name Message API: {{.APIFile}}, pOrg: {{.ProviderOrg}}: Updated catalogs: property: {{.OldCatalogName}} with catalog name: {{.NewCatalogName}} based on matching catalog title: {{.FoundCatalogTitle}} since API Gateway requires catalog name instead of catalog title in `catalogs:`**

**Explanation**

In v5, the `catalogs:` property uses names based on the Catalog Title. In API Gateway, the Catalog Name must be used. The Migration Utility attempted to find a matching Catalog Title in the same provider-org as the API, and succeeded. The `catalogs:` property that formerly used the Catalog Title has been automatically updated by the Migration Utility to use the Catalog Name of the Catalog instead.

**Response**

Review your API to confirm that the Catalog Name used in the `catalogs:` property is correct.

**Severity**

info

## Miscellaneous messages

---

- [UT01](#)
- [UT02](#)
- [UT03](#)
- [UT04](#)
- [UT05](#)
- [UT06](#)
- [UT07](#)
- [UT08](#)
- [UT09](#)
- [UT10](#)
- [UT11](#)
- [UT12](#)
- [UT13](#)
- [UT14](#)
- [UT15](#)
- [UT16](#)

**UT01 Invalid suffix provided. Using `-apigw` as the default for API Gateway API `title` and `x-ibm-name`.**

**Explanation**

The provided suffix input was incorrectly formatted. The migration utility uses `-apigw` as the default suffix. It will be appended to the end of the `x-ibm-name` and `title` to maintain uniqueness from the original v5-compatible input API.

**Severity**

warn

**UT02 No API suffix provided. Using `-apigw` as the default for the API Gateway API `title` and `x-ibm-name`.**

**Explanation**

The `--suffix` option is optional. Because no input was provided, the migration utility defaults to `-apigw`. This suffix will be appended to the end of the `x-ibm-name` and `title` to maintain uniqueness from the v5-compatible input API.

**Severity**

info

**UT03 Input file not found: ` %s `, exiting. Verify that the v5-compatible YAML input file exists and is a valid OpenAPI.**

**Explanation**

See message text.

**Severity**

error

**UT04 An error occurred while reading the input file: ` %s `. Verify that the v5-compatible YAML input file exists and is a valid OpenAPI.**

**Explanation**

See message text.

**Severity**

error

**UT05 Output directory not found: ` %s `, creating it.**

**Explanation**

See message text.

**Severity**

info

**UT06 An error occurred while trying to create the output directory: ` %s `. Verify that this utility has sufficient access to the filesystem in order to write to the output directory and file.**

**Explanation**

See message text.

**Severity**

error

**UT07 Finished creating API Gateway API draft from v5-compatible API input file. Attempting to save to output file.**

**Explanation**

See message text.

**Severity**

info

**UT08 Output directory not found: ` %s `, exiting. Verify that this utility has sufficient access to the filesystem in order to write to the output directory and file. For further assistance, contact IBM Support.**

**Explanation**

See message text.

**Severity**

error

**UT09 An internal error occurred: ` %s `.**

**Explanation**

See message text.

**Severity**

error

**UT10 An internal error occurred processing the condition map: ` %s `.**

**Explanation**

See message text.

**Severity**

error

**UT11 Output API Gateway draft API saved as: ` %s `.**

**Explanation**

See message text.

**Severity**

info

**UT12 Input v5-compatible API could not be converted. Exiting.**

**Explanation**

See message text.

**Severity**

error

**UT13 An error occurred while processing the v5-compatible input API.**

**Explanation**

See message text.

**Severity**

error

**UT14 An error occurred while converting the v5-compatible input API to API Gateway: ` %s `.**

**Explanation**

See message text.

**Severity**

error

**UT15 An error occurred adding a comment to the resultant YAML: ` %s `.**

**Explanation**

See message text.

**Severity**

error

**UT16 An error occurred during v5 escaped character emulation parsing the updated source: %s"**

**Explanation**

Contact IBM support.

**Severity**

error

---

## Referenced objects

**OAuth2 messages**

- [OA201](#)
- [OA202](#)

**OA201 Found OAuth2 security definition using the provider ` %s `.** The corresponding OAuth provider for type API Gateway must exist on the server.

**Explanation**

See message text.

**Severity**

info

**OA202 Found OAuth2 security definition with no provider specified. The API Gateway OAuth2 security definition requires a provider to be specified in an `x-ibm-oauth-provider` field.**

Explanation

See message text.

Severity

error

### TLS-Profile messages

- [TLS01](#)
- [TLS02](#)

**TLS01 Renamed `api-sslcli-all` `tls-profile` to `api-sslcli-client-all` to conform to the API Gateway specification.**

Explanation

See message text.

Severity

info

**TLS02 Added `1.0.0` as `tls-profile` version to `%s` to conform to API Gateway specification.**

Explanation

The API Gateway TLS profiles support versioning. A default version of `1.0.0` has been added to the profile configuration.

Severity

info

---

## Parse actions added to the API assembly after migration

The migration utility command `apicm archive:port-to-apigw` adds parse actions to the API assembly depending on the policies specified in your migration configuration.

Review the API assembly to determine if any of the parse actions or other actions added by the migration utility can be removed. For more information, see [Fine-tuning APIs after migration](#).

Parse actions are added to the assembly according to the rules in the following tables.

Table 1. Parse actions added to the API assembly after conversion of policies to 1.0.0 or 2.0.0

| Policy                                                           | Where the parse action is added                                                 |
|------------------------------------------------------------------|---------------------------------------------------------------------------------|
| If a <b>POST, PUT, DELETE, OPTIONS, or PATCH</b> method is found | As the first action in the assembly                                             |
| invoke 2.0.0                                                     | After the invoke action, except when invoke is the last action in the assembly. |
| json-to-xml 2.0.0                                                | Before the json-to-xml action                                                   |
| proxy 2.0.0                                                      | After the proxy action                                                          |
| validate 2.0.0                                                   | Before the validate action                                                      |
| xml-to-json 2.0.0                                                | Before the xml-to-json action                                                   |
| xslt 1.0.0 or 2.0.0                                              | Before the xslt action                                                          |

Table 2. Parse actions added to the API assembly after conversion of invoke and proxy policies to 1.5.0

| Policy       | Where the parse action is added                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| invoke 1.5.0 | A parse action is added after the invoke if any of the following are true: <ul style="list-style-type: none"><li>• <code>stop-on-error</code> is specified</li><li>• <code>final-action</code> is set to <code>false</code>. This is the default setting.</li><li>• <code>x-ibm-gateway-optimize-invoke</code> is explicitly set to <code>false</code></li></ul> The parse action is added because there can be a subsequent use of the output. |
| proxy 1.5.0  | A parse action is added after the proxy if <code>final-action</code> is set to <code>false</code> .                                                                                                                                                                                                                                                                                                                                             |

---

## Fine-tuning APIs after migration

After running the migration utility command `apicm archive:port-to-apigw`, review the converted APIs to determine if any of the changes made by the migration utility can be modified to enhance performance and usability.

If the migration utility makes any of the following changes to your APIs, consider making the suggested changes.

- If the last policy in the assembly has output that is set to a variable that is not used afterward, remove the output set or the entire policy.
- Remove any unnecessary parse actions that the migration utility adds to the assembly. For information about the parse actions added by the migration utility, see [Parse actions added to the API assembly after migration](#).
- If the assembly contains redact policies, verify that these policies are necessary. Delete any unnecessary redact policies.

---

## Frequently asked questions about migrating APIs to API Gateway

Answers to some of the most frequently asked questions about migrating APIs to API Gateway.

- [What is port-to-apigw?](#)
- [When is it necessary to run the port-to-apigw command, and how do I use it?](#)
- [Why is the port-to-apigw command needed for migrating to v10 API Gateway?](#)
- [What is the difference between these terms: "porting to API Gateway", "port-to-apigw", and "migration"?](#)
- [What is v5 policy emulation and how does it work?](#)
- [What is the purpose of the policy parameter options in port-to-apigw?](#)
- [How do the v5 emulated policies invoke\\_1.5.0, proxy\\_1.5.0, xslt\\_1.0.0, and gatewayscript\\_1.0.0 work?](#)
- [What is the difference between the 1.5.0 and 2.0.0 versions of the invoke and proxy policies, and the 1.0.0 and 2.0.0 versions of the gatewayscript policy? Why isn't the 2.0.0 version the default version?](#)
- [Should I rewrite my GatewayScript policies?](#)
- [What is the optimize-gws option?](#)
- [What is the gateway-extension.zip file and how does it work?](#)
- [How does the migration utility migrate custom policies?](#)
- [Should I set the custom-policies-scope option to global or catalog?](#)
- [Why are my custom policies not showing up in the Assembly Editor policy palette?](#)

What is **port-to-apigw**?

**port-to-apigw** is a command in the API Connect migration utility that converts all APIs, products, subscriptions, OAuth providers, and custom policies into v10 native API Gateway artifacts. These newly created artifacts live side-by-side with the original v5-compatible artifacts, and have **-apigw** appended to the end of the filename. Starting in 10.0.5.4 a porting suffix can be used to override **-apigw**.

For more information, see [Converting APIs to DataPower API Gateway](#).

When is it necessary to run the **port-to-apigw** command, and how do I use it?

Run this command if you are migrating to a v10 native API Gateway. Follow the migration steps that are appropriate for your v5 environment. For instructions on running the **port-to-apigw** command, see [Migration steps](#). For information on configuring migration options, see [Configuring migration options for DataPower API Gateway](#). For other information to help you optimize migration, see [Migration considerations](#).

Why is the **port-to-apigw** command needed for migrating to v10 API Gateway?

The required fields, properties, settings, and configuration in v10 native API Gateway differs from v5-compatible mode. The **port-to-apigw** step in the migration process is needed to convert the artifacts to conform to the changes.

What is the difference between these terms: "porting to API Gateway", **port-to-apigw**, and "migration"?

"Porting to API Gateway" describes what the **port-to-apigw** command does in the migration utility. "Migration" is the overall purpose of the migration utility. The **port-to-apigw** step is necessary for migrating to a v10 native API Gateway. Migration can be completed without the **port-to-apigw** step if you are migrating to a v10 deployment that uses only the v5-compatible DataPower Gateway.

What is v5 policy emulation and how does it work?

The v5 policy emulation layer allows both custom policies and the 1.0.x policies to run on the native v10 API Gateway. It is deployed automatically by the API Connect migration utility to the `local:///isp` folder of the DataPower appliance when a custom policy is found in the v5 backup, or if you specify a 1.0.x policy in the policy option for the **port-to-apigw** command. The deployment is first staged by the **port-to-apigw** command, and pushed by the **archive:push** command. Follow the steps as outlined in [Migration steps](#).

Note that 1.5.x policies do not use v5 policy emulation, and exist only as native wrappers to the 2.0.0 implementation.

What is the purpose of the policy parameter options in **port-to-apigw**?

Input options allow you to customize your migration. If you want to use the native policies, such as `invoke_2.0.0` instead of the default `invoke_1.5.0-compatible` option, you can select the appropriate option. However, using these options can result in migration errors that require manual changes to get your APIs up and running. If you want to be up and running quickly, it is best to use the default options for the input. The default options convert APIs in a way that is most backwards compatible with v5.

For more information about the migration configuration options, see [Configuring migration options for DataPower API Gateway Configuration parameters for migrating v5 APIs](#). For explanations of errors and other messages generated during migration and suggestions for manual changes to APIs, see [Messages generated during conversion of APIs to DataPower API Gateway](#).

How do the v5 emulated policies `invoke_1.5.0`, `proxy_1.5.0`, `xslt_1.0.0`, and `gatewayscript_1.0.0` work?

When the push step of the migration utility is completed, these policies will exist on the DataPower Gateway as assembly function objects. The 1.0.x policies call the v5 policy emulation layer in order to function as if they were v5 policies. The 1.5.x policies are wrapper policies to the 2.0.0 implementation. They call the 2.0.0 policies and exist only as a compatibility layer between the original v5 policy and the new 2.0.0 policy.

What is the difference between the 1.5.0 and 2.0.0 versions of the `invoke` and `proxy` policies, and the 1.0.0 and 2.0.0 versions of the `gatewayscript` policy? Why isn't the 2.0.0 version the default version?

The 1.5.x policy implementations are native wrappers to the underlying 2.0.0 implementation. They exist only to create a compatibility layer between the v5 policy and the APiGW native policy. The `gatewayscript_1.0.0` policy uses v5 policy emulation to run as if it were on a v5 environment. The performance of the 1.0.x v5 emulation layer policies in API Gateway is the same or slightly better than the v5 and v5-compatible versions. The performance of 1.5.x policies is slightly slower than their corresponding 2.0.0 policies. You can inspect the 1.5.x wrapper implementation in Assembly Functions in DataPower Gateway.

Should I rewrite my GatewayScript policies?

Most GatewayScript policies do not require rewriting, as the native v10 API Gateway contains an enhanced compatibility layer that accepts most of the **apim** functions without modification. In most cases, GatewayScript policies require rewriting when they contain any of the following:

- inline parameters
- `require('service-metadata')` and service variables associated with it
- `require('gmt')`
- `require('fs')`
- OAuth context variables

What is the `optimize-gws` option?

The `optimize-gws` migration option converts certain v5 gateway script functions to the native API Gateway implementation. This setting is optional because the compatibility layer already handles processing of original v5 gateway script functions. If set to true, the migration utility makes the following conversions:

- `apim.getvariable()` is converted to `context.get()`
- `apim.setvariable()` is converted to `context.set()`
- `hm.current.remove()` is converted to `context.clear()`

What is the `gateway-extension.zip` file and how does it work?

After running the `port-to-apigw` command, a `gateway-extension.zip` file is located in the `cloud/admin-org/availability-zones/default-availability-zone/gateway-services/<gateway-service>` folder. This file is automatically generated by the migration utility during the processing of the `port-to-apigw` command. It contains a `manifest.json` file, and every custom policy under the gateway service that came from the v5 backup. The `manifest.json` file contains a list of custom policies to push, and backwards-compatible policies to advertise and make available for use by APIs. These backwards-compatible policies include `invoke_1.5.0`, `proxy_1.5.0`, `xslt_1.0.0`, and `gatewayscript_1.0.0`. During the `archive:push` steps of the migration process, this zip file is pushed to the API Manager as a gateway extension.

For more information, see [Gateway extensions manifest](#).

How does the migration utility migrate custom policies?

Custom policies are migrated in multiple steps, depending on whether the `custom-policies-scope` option is set to `global` or `catalog`. Once custom policies are pushed to the DataPower Gateway, they use the `v5 policy emulation` layer that was unpacked into the `local:///isp` folder in order to run on the native v10 API Gateway.

When `port-to-apigw` is run, the following steps are performed depending on the `custom-policies-scope` setting.

- If `custom-policies-scope` is set to `global`, and there are no duplicate custom policies with differing implementations, the custom policies under each gateway service are bundled in a `gateway-extension.zip` file. This zip file contains a `manifest.json` file that describes the contents. The `gateway-extension.zip` file is copied to the `admin-org gateway-service` folder. These steps are performed automatically when `port-to-apigw` is run. Next, you must push the `admin-org` during the `archive:push` step, which pushes the `gateway-extension.zip` bundles (one per gateway) to the API Manager using an internal `APIC Toolkit gateway-extensions:create` call. The custom policies within the bundle are advertised on the API Manager. The custom policies and compatible legacy policies are shown in the policy pallet in the API Manager Assemble UI. These custom policies are also in the DataPower filesystem in the `local:///` folder. Note that if duplicate custom policies with the same name, version, and differing underlying implementation are found, the migration utility logs a message and automatically falls back to `catalog` scope to prevent the migration from failing.
- If `custom-policies-scope` is set to `catalog`, the custom policies under each gateway service are bundled in a `gateway-extension.zip` zip file. This zip file contains a `manifest.json` file that describes the contents. The `gateway-extension.zip` file is then copied to the `admin-org gateway-service` folder. The custom policy definition files under each catalog are copied into the `catalog-scoped-staging` folder, where their gateway type is set to `datapower-api-gateway`. Next, you must push the `admin-org` during the `archive:push` step, which pushes the `gateway-extension.zip` bundles (one per gateway) to the API Manager using an internal `APIC Toolkit gateway-extensions:create` call. Note that the custom policies and compatible legacy policies are not visible in the policy pallet in the UI. You can see the custom policies in the DataPower filesystem in the `local:///` folder. Before you can publish APIs that use these policies, you must push your `provider-orgs` during the `archive:push` step. The custom policy definitions as described earlier from the `catalog-scoped-staging` folder are then pushed to the API Manager automatically using an internal `APIC Toolkit policies:create` call. The APIs containing the newly migrated catalog-scoped custom policies are then published.

Note: The folders and zip files are created automatically, and you should not need to modify them. For instructions on the tasks you must perform, see [Migration steps](#).

Should I set the `custom-policies-scope` option to `global` or `catalog`?

Global scope custom policies are visible to every catalog and org on the gateway. Catalog scope custom policies are visible only to the catalog that they were migrated from. If you plan to migrate to v10 with the intention of custom policy availability the same as it is on v5, then use the default `catalog` setting.

Why are my custom policies not showing up in the Assembly Editor policy palette?

See the following troubleshooting information: [Custom policies not appearing in the policy palette after migration](#)

---

## Migration mapping files

You can use mapping files to map the name of a v5 object to a corresponding v10 object that has a different name.

The API Connect migration process allows you to make modifications to the configuration information after it has been extracted from the source v5 system and before it is uploaded to the target v10 system.

The names of some structures may not match exactly between v5 and v10. Since the names of some high-level objects may be intentionally different in the v10 environment, you can substitute values so that references to v5 object names (user registries, gateway services etc.) can be bound to the newly created objects in v10.

Note: If the names are preserved exactly between the v5 and v10 environments, you do not need to map any names.

- [What is a mapping file?](#)
- [Mapping a gateway service](#)
- [Mapping a migrated OAuth shared secret key](#)
- [Mapping SNI settings](#)
- [Mapping a provider organization](#)

---

## What is a mapping file?

What does a mapping file do?

Mapping files are simple yaml files that may be placed at any level in the directory structure produced by the `apicm archive:unpack` command where there is a yaml file describing a configuration object. A mapping file contains yaml `key: value` pairs that provide the values to apply in place of the original configuration values.

- You can use mapping files to override any values within the `.yaml` files.
- Common mappings can be specified in `mapping.yaml`.
- Server specific mappings can be provided in `mapping.<servername/ip>.yaml`. For example: `mapping.hostname.com.yaml` for `hostname.com` server.

What kinds of settings can I map?

You can override any values within a `.yaml` files. The most common mappings for v5 migrations are:

- Gateway service names
- Provider organization names
- User registry names

What does a mapping file contain?

A mapping entry consists of two or three lines:

- The first line of the mapping entry should be the file name you want to override, without the `.yaml` extension. For example, when overriding a value in `provider-org.yaml`, the first line of the mapping is `provider-org`.
- Additional lines have `key: value` pairs that override `key: value` pairs found in the main `.yaml` file. The most common usage is for overriding the name. To do this include the `name:` setting, followed by the new value for v10. For example, if you have a v5 provider organization `myv5porg`, and you want to map it to `myv10porg`, the second line is `name:`

```
provider-org:
 name: myv10porg
```

- There are a few exceptions for some additional fields with special meaning in the mapping file for gateway services. See [Mapping a gateway service](#).

Where do I create mapping files

Examine the `/cloud` directory hierarchy that you created when you ran `apicm`

`archive:unpack`. Locate the directory that has the `.yaml` file with the v5 value you want to overwrite. In that same directory, create a new file named `mapping.yaml`. For a detailed example, see the [Mapping tutorial](#).

## Mapping a gateway service

You can use a mapping file to tell the migration program the name of the target gateway service in v10 to use when migrating from v5 gateway service data to v10 gateway service data. You might want to map a v5 gateway service name to a v10 gateway service if the names do not match, or if the number of gateway services does not match.

Note: You can map a v5 gateway service to a v10 DataPower API Gateway. In this case, the mapping definition requires a `gateway-type` field. See the [Mapping tutorial](#). After you unpack the v5 data, you can find each of v5 gateway services in `cloud/admin-org/availability-zones/availability-zone-default/gateway-services/`.

For example, the following `cloud/admin-org/availability-zones/availability-zone-default/gateway-services/` hierarchy shows entries for three v5 gateways named: `gateway-2536`, `gateway-2537`, and `gateway-2538`.

Figure 1. Example admin-org with entries for three v5 gateway services

```
admin-org
├── availability-zones
│ ├── availability-zone-default
│ │ ├── availability-zone.yaml
│ │ └── gateway-services
│ │ ├── gateway-2536
│ │ │ ├── gateway-service.yaml
│ │ │ └── gateway-service.yaml
│ │ ├── gateway-2537
│ │ │ ├── gateway-service.yaml
│ │ │ └── gateway-service.yaml
│ │ └── gateway-2538
│ │ ├── gateway-service.yaml
│ │ └── gateway-service.yaml
```

## Mapping a migrated OAuth shared secret key

Information about migrated OAuth shared secret keys is contained in your `gateway-service.yaml` file. When you map a gateway to a new name, you must ensure that the OAuth shared secret key is associated with mapped gateway name.

Note: Make sure you only migrate one OAuth shared secret key per gateway service. You will get an error if you attempt to map more than one to the same gateway service.

1. Determine if your `gateway-service.yaml` includes information for an OAuth Shared Secret Key (`oauth_shared_secret:`). For example:

```
api_version: 2.0.0
metadata:
 v5_created_at: 2018-12-01 17:43:03.065 +0000 UTC
 v5_created_by: admin
 v5_id: 5c02c827e4b0e816d3874747
 v5_updated_at: 2018-12-04 15:59:03.282 +0000 UTC
 v5_updated_by: admin
name: gw3-cf31
oauth_shared_secret: 0xF2DB3AE67A7988C06FF1EBA5C2B51F04ABC5D3D1192B4C994CA00C79FAB5ADAD
type: gateway_service
```

2. If `oauth_shared_secret:` is found, add `migrate_oauth_shared_secret:` `true` to the `mapping.yaml`. For example:

```
gateway-service:
 name: gateway-10-1
 migrate_oauth_shared_secret: true
```



## Mapping SNI settings

You can include Server Name Indication (SNI) settings in the mapping file. When enabled, these settings, which include the host and TLS Server Profile pairs, are included in the migration of gateway services.

Note: You can migrate only one set of SNI settings for a v10 gateway service. If you are mapping multiple v5 gateway services to a single v10 gateway service, only the first set of SNI settings is migrated and warnings are generated for the additional mappings.

1. Determine whether your `gateway-service.yml` file contains SNI settings to migrate, as in the following example.

```
api_version: 2.0.0
gateway_service_type: datapower-gateway
metadata:
 v5_created_at: 2022-02-02 20:07:29.566 +0000 UTC
 v5_created_by: admin
 v5_id: 61fae481e4b0b16dad38691a
 v5_updated_at: 2022-02-02 20:07:29.566 +0000 UTC
 v5_updated_by: admin
name: gateway-service6-fabd
oauth_shared_secret: 0x014D004DC0A32EF091C17FEFCF10A621C36A92CE88BCE0F8FC2463B096D72D99
sni:
- host: internal.apic.ibm.com
 tls_server_profile_url: idurl://../../../../../../../../tls-server-profiles/gatewaytlsprofile-1.0.0.yml
- host: partner.apic.ibm.com
 tls_server_profile_url: idurl://../../../../../../../../tls-server-profiles/default-ssl-profile-1.0.0.yml
type: gateway_service
```

2. If SNI settings are found, add `migrate_sni: true` to the `mapping.yml`. For example:

```
gateway-service:
 name: gateway-10-1
 migrate_sni: true
```

## Mapping tutorial

You create and bring up a new gateway service in v10 called `gateway-10-1`. You want the data (e.g Catalogs) associated with the v5 gateway `gateway-2536` to be migrated to the v10 gateway named `gateway-10-1`.

1. Go to the `/cloud` hierarchy and locate the `gateway-2536/` directory.

```
|
| └─ gateway-services
| └─ gateway-2536
| └─ gateway-service.yml
```

2. Open the `gateway-service.yml` and verify that the `name:` entry specifies the v5 gateway service that you want to map.

```
api_version: 2.0.0
v5_id: 5c0icef4e45816d32290fb
name: gateway-2536
type: gateway-service
```

3. To map `name: gateway-2536` to a new value, use a text editor to create a **new** file named `mapping.yml`.

- a. On a new line, enter `gateway-service:`.

How do I know what to enter on the first line?

Enter the name of the file that you are overriding, without the `.yml` suffix. In this case, you are overwriting `gateway-service.yml`, so you enter `gateway-service:`.

- b. On the next line, indent two spaces (as required for valid yaml), and enter `name:`  
`gateway-10-1`.

```
gateway-service:
 name: gateway-10-1
```

How do I know which name to use when mapping the v10 gateway-service?

In the v10 Cloud Manager UI, go to the Topology page. In the Default Availability Zones table, locate the Service column and click the gateway service entry. On the Edit DataPower Gateway Service page, in the Gateway Details section, locate the value in the **Name** field. Use this value.

Note: Do not use the value in the **Title** field. The value in the **Name** field is slugged to contain only characters that are supported in v10, and thus might differ from the value in the **Title** field.

- c. If the new v10 gateway service `gateway-10-1` is a DataPower API Gateway, add a third line to specify the gateway type:

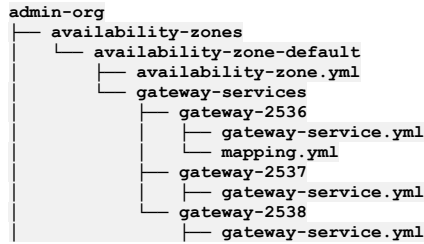
```
gateway-service:
 name: gateway-10-1
 gateway-type: datapower-api-gateway
```

If the new v10 gateway service `gateway-10-1` is a v5-compatible gateway, do not specify `gateway-type`. When `gateway-type` is not specified, the mapping defaults to v5-compatible.

- d. The mapping is complete. Save the file in the same directory.

The following diagram shows the location of the new `mapping.yml` file in the `c1oud/` hierarchy for `admin-org`. Note that it is in the same directory as the `gateway-service.yml` file for the v5 service it applies to (`gateway-2536`).

Figure 2. Example `admin-org` with `mapping.yml` for `gateway-2536` gateway service entry



When the mapping utility (`apicm archive:push`) runs, it automatically uses the new `name`: value found in `mapping.yml` to overwrite the `v5` value found in `gateway-service.yml`. Note that the gateway service on v10 must be up and running

Do I have to edit the existing `gateway-service.yml`?

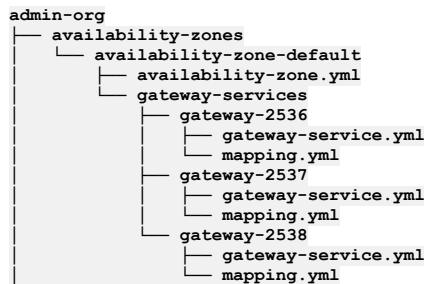
No, just create a new `mapping.yml` file and add it to the same directory as `gateway-service.yml`.

What if I want to map several v5 gateway-services to one v10 gateway-service?

If you have several v5 gateway services, and you want to map each of them to the same v10 gateway service, `gateway-10-1`, you can copy your newly edited `mapping.yml` into the directory for each gateway service. Under each gateway entry, place a copy of your new `mapping.yml`.

The following diagram shows the locations of the three copies of `mapping.yml` files (one each for `gateway-2536`, `gateway-2537`, and `gateway-2538`) in the `cloud/` hierarchy for `admin-org`.

Figure 3. Example `admin-org` with `mapping.yml` files for three v5 gateway services



## Mapping a provider organization

The `apicm archive:unpack` command produces a directory structure that includes a subdirectory for each provider organization in the original system. A provider organization named `onlinebanking` would be represented by a directory name `cloud/provider-orgs/onlinebanking`, containing a file named `provider-org.yml`:

```

api_version: 2.0.0
name: onlinebanking
org_type: provider
state: enabled
title: OnlineBanking
summary: migrated onlinebanking
metadata:
 v5_id: 5c0ebfdae4b007df531ffe40
 ...

```

To override the organization name (indicated by the `name` property), create a file named `mapping.yml` in the same directory as the `provider-org.yml` file, and include the new name to apply to the organization. Example `mapping.yml`:

```

provider-org:
 name: digitalbanking

```

Here, the result of the override is to cause a provider organization named `digitalbanking` to be used as the target provider organization for the contents originally extracted from the `onlinebanking` organization.

## Migrating OAuth providers and OAuth secured APIs

You can migrate v5 OAuth providers and OAuth secured APIs to your v10 API Connect deployment.

In v5 OAuth providers are a specialized type of API that are published within products. In v10 OAuth providers are first-class objects that are directly published in the provider organization. This is applicable for both DataPower Gateway (v5-compatible) and DataPower API Gateway.

Also, v10 adds tight coupling between the API using OAuth and the OAuth provider object. Hence an API using OAuth security must link to the corresponding OAuth provider object. For a review of how API Connect v10 uses OAuth providers, see [OAuth Provider overview](#).

The API Connect Migration Utility (AMU) migrates all OAuth provider APIs from v5 to the first-class OAuth provider objects on both DataPower Gateway (v5-compatible) and DataPower API Gateway.

In addition, the AMU uses logic to determine which OAuth provider is used by a given API and populates it in the newly added field `x-ibm-configuration` of API security definitions. If the AMU is unable to find a match between an API using OAuth and the OAuth provider, it prints out error messages. See [Error messages and warnings from archive:unpack](#).

- To ensure that APIs that use OAuth security definitions are mapped to the corresponding OAuth provider APIs in v10, make sure that either the Authorization URL or the Token URL defined in OAuth Security Definitions of APIs that use OAuth correctly point to the base path of the OAuth provider API. Note that the AMU uses logic to map consumer APIs to provider APIs, and adds `x-ibm-oauth-provider` in the OAuth Security Definition of the consumer API.

- To verify whether OAuth provider APIs are migrated correctly, ensure that all the APIs that use OAuth Security Definition have the new field `x-ibm-oauth-provider` added. Check the value of that field and make sure that it points to a correct OAuth provider resource object that was created and configured in the catalog to which the API is published.
- As mentioned earlier, in v5 OAuth providers are a specialized type of API that are published within products. In v10 OAuth providers are instead first-class objects that are directly published in the provider organization, and are not APIs that are published within products. Because of this, it is possible that you may have some v5 products that only contained OAuth provider APIs. Since these products do not contain any Open API definitions in v10, they will not be published as all products in v10 must contain at least one Open API definition.
- Because v10 OAuth providers are first-class objects rather than APIs as in v5, the application authentication feature is not available to v10 OAuth providers migrated from v5.

## Migrating OAuth token revocation information

In v5, token revocation could be enabled either using native token management system (TMS) or using external TMS. For migration to v10:

- Native TMS migration
  - DataPower Gateway (v5-compatible)
 

If you had configured native token revocation, then you would have configured Quota enforcement peering group objects on v5 to synchronize between gateway servers. When migrating to v5-compatible, a QES peering group can contain QES peers from both v5 Multi-Protocol Gateway Service (MPGW) and v10 DataPower Gateway, so the revocation information is retained.

After setting up a v10 DataPower Gateway instance, follow these steps to configure the v10 DataPower Gateway quota enforcement server (QES) to use the same QES on both v5 and v5-compatible. You must configure the QES and verify that it is enabled and functional before migrating from v5 to v10. For more details about configuring the QES, see the DataPower Gateway documentation topic *Configuring the quota enforcement server*.

    1. Log into the graphical interface of the DataPower Gateway instance in the default domain.
    2. Cluster the v5 MPGW QES with the v10 DataPower Gateway QES.
      - a. In DataPower Gateway, navigate to Objects > System Settings > Quota Enforcement Server.
        - b. Specify the password alias, data storage location, and the server and monitor port numbers.
 

If your v10 DataPower Gateway is a Docker instance, configure the QES to store data in memory. For more information about RAID and Docker, see the DataPower Gateway documentation topic *RAID and a DataPower Docker image*.
        - c. Enable Peer group mode.
        - d. Specify the IP address of the DataPower Gateway.
 

The IP address can be the IP address on any interface of the current appliance and must be accessible by other peers in the peer group.
        - e. Specify the IP addresses of the v5-compatible gateways to add them as peers.
 

The appliance connects to each peer in the order in which the peers are added in the list.
        - f. Apply and save the configuration.
      3. As a test, revoke a token on the v5 MPGW server.
 

Revocation data will replicate to v10 DataPower Gateway QES.
      4. Migrate the v5 APIs to the v10 DataPower Gateway QES.
      5. Send a request to the v10 DataPower Gateway QES with previously revoked token, which should fail as expected.
 

The expected failure indicates that the APIs were migrated and revocation data was replicated.
    - DataPower® API Gateway
 

When migrating to DataPower API Gateway, QES is not used for native token management and revocation. Instead, you can migrate token revocation information from a v5 QES to a gateway-peering cache in v10 DataPower API Gateway. For information, see [Migrating token revocation information to DataPower API Gateway](#).
- [Migrating token revocation information to DataPower API Gateway](#)

You can use the Token Migration Utility (TMU) to migrate token revocation information from a v5 quota enforcement server (QES) to a gateway-peering cache in v10 DataPower API Gateway.
- [Token migration errors](#)

After you run the Token Migration Utility (TMU), check the log in the logs subdirectory for errors. If it contains any of the following error messages, complete the troubleshooting steps before you run the utility again.

## Migrating token revocation information to DataPower API Gateway

You can use the Token Migration Utility (TMU) to migrate token revocation information from a v5 quota enforcement server (QES) to a gateway-peering cache in v10 DataPower® API Gateway.

### Before you begin

Download the TMU from [IBM Fix Central](#). The TMU includes a container image and a separate configuration file.

The TMU requires the following versions of DataPower Gateway on the source and destination servers.

| System      | Required DataPower Gateway versions |
|-------------|-------------------------------------|
| Source      | v10.5.0.0 or higher                 |
|             | v10.0.4.0sr1 or higher              |
|             | v10.0.1.4 or higher                 |
|             | v2018.4.1.16 or higher              |
| Destination | v10.5.0.4 or higher                 |

## About this task

You can use the TMU to migrate revoked access tokens, revoked refresh tokens, resource owner consent, and, optionally, revoked resource owner consent.

Token migration consists of the following stages:

- Export QES data to a JSON file in the local directory.
- Import QES data into v10 DataPower Gateway from the JSON file.
- Configure a deny list in the v10 token cache.

The TMU transforms the data from QES to v10 format. It uses the DataPower Gateway REST management interface to expose the data and import it to the gateway-peering instance. When TLS is enabled in the gateway-peering instance, the TMU creates a TLS proxy and a TLS client profile to import the QES data. When TLS is not enabled, the TMU creates a TCP proxy to import the data.

Attention: Verify that the following requirements are met before you run the TMU.

- Ensure that the DataPower Gateway REST management interface is enabled. For information about enabling the REST management interface and configuring gateway peering for DataPower API Gateway, see [Configuring DataPower API Gateway](#).
- Gateway peering requirements
  - The v10 gateway peering password must be defined.
  - The gateway-peering instance in DataPower Gateway for the internal token store requires that data persists across a restart to prevent data loss. Ensure that this instance is not configured to store data in memory.
- v5 QES configuration requirements
  - Peer group mode must be disabled.
  - The listening server port must use the default value `127.0.0.1:16379`.
  - The v5 QES password must be defined.

Tip:

- The `./tmu` command uses the argument `--stage` or `-s` to indicate the stage of migration to perform.
- Use the `--help` argument to get information about any of the token migration stages.

## Procedure

1. Define the following fields in the `config.json` file. All fields must be present and defined unless otherwise noted.

Table 1. Token migration configuration options

| Field                                           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>v5-gateway-address</code>                 | v5 DataPower Gateway address                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>v5-gateway-username</code>                | v5 DataPower Gateway username                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>v5-gateway-password</code>                | v5 DataPower Gateway password                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>v5-qes-password</code>                    | v5 QES password                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>v5-cache-proxy-port</code>                | The user-defined port on which to expose the v5 QES proxy. Ensure that this port is not in use.                                                                                                                                                                                                                                                                                                                                                           |
| <code>v5-cache-container-port</code>            | If DataPower Gateway is deployed in Docker, the port that is exposed on the container that maps to the port defined in <code>v5-cache-proxy-port</code> . If DataPower Gateway is not deployed in Docker, this field can be undefined. When the cache container port is defined, the TMU attempts to connect to these ports, assuming that DataPower Gateway is hosted in a container. When it is not defined, the TMU connects to the cache proxy port.  |
| <code>v5-rest-management-interface-port</code>  | REST management interface port. If DataPower Gateway is deployed in Docker, enter the container port.                                                                                                                                                                                                                                                                                                                                                     |
| <code>v10-gateway-address</code>                | v10 DataPower Gateway address                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>v10-gateway-username</code>               | v10 DataPower Gateway username                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>v10-gateway-password</code>               | v10 DataPower Gateway password                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>v10-gwp-password</code>                   | v10 gateway-peering password                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>v10-cache-gateway-peering-name</code>     | Gateway-peering object name that is mapped to the v10 token cache. If peer group mode is enabled, ensure that the specified gateway-peering object is the primary member, not a replica.                                                                                                                                                                                                                                                                  |
| <code>v10-cache-proxy-port</code>               | The user-defined port on which to expose the v10 token cache proxy. Ensure that this port is not in use.                                                                                                                                                                                                                                                                                                                                                  |
| <code>v10-cache-container-port</code>           | If DataPower Gateway is deployed in Docker, the port that is exposed on the container that maps to the port defined in <code>v10-cache-proxy-port</code> . If DataPower Gateway is not deployed in Docker, this field can be undefined. When the cache container port is defined, the TMU attempts to connect to these ports, assuming that DataPower Gateway is hosted in a container. When it is not defined, the TMU connects to the cache proxy port. |
| <code>v10-rest-management-interface-port</code> | REST management interface port. If DataPower Gateway is deployed in Docker, enter the container port.                                                                                                                                                                                                                                                                                                                                                     |
| <code>export-file-name</code>                   | File name for extracted v5 data.                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>log-file-name</code>                      | Log file name. After any of the TMU commands are run, a log file is generated in a logs subdirectory of the TMU directory. Log files are named according to the format <code>DATE-TIME %log-file-name%</code> .                                                                                                                                                                                                                                           |
| <code>log-level</code>                          | Log level to filter logged messages by severity. Valid values are <code>debug</code> , <code>error</code> , or <code>info</code> .                                                                                                                                                                                                                                                                                                                        |
| <code>import-revoked-consent</code>             | Whether to import consent revocation entries. Valid values are <code>true</code> or <code>false</code> .                                                                                                                                                                                                                                                                                                                                                  |

2. Start the TMU container.
3. Copy the edited `config.json` file into the `/home/tmu/ibm` directory of the TMU container.
4. Run the following command to export QES data to a JSON file in the local directory.

```
./tmu --stage v5-export
```

The QES data is exported to a JSON file in the same directory as the TMU. The name of the exported file is specified in the `export-file-name` configuration field. Each exported entry in the file has the following fields.

```
timeExport
 Unix epoch timestamp.
value
```

String value to the key in the QES.  
`ttl`  
Time to live of the entry in seconds.

Important: Do not modify the entries in the export file.

5. Run the following command to import QES data from the exported JSON file into v10 DataPower Gateway.

```
./tmu --stage v10-import
```

6. Run the following command to configure a deny list in the gateway-peering instance.

```
./tmu --stage token-cache-mode-set-deny
```

Attention: When the gateway-peering token cache mode is set to deny-based caching, it cannot be reset to allow-based caching.

7. Optional: Run the following command at any time to view the cache mode configuration in the v10 token cache.

```
./tmu --stage token-cache-mode-view
```

## What to do next

Check the log file in the logs subdirectory for errors. For information about common errors and troubleshooting steps, see [Token migration errors](#).

## Token migration errors

After you run the Token Migration Utility (TMU), check the log in the logs subdirectory for errors. If it contains any of the following error messages, complete the troubleshooting steps before you run the utility again.

Cache at {address:port} is not responsive. The proxy may be down or using an incorrect remote port.

The proxy to either the QES or gateway-peering token cache is not responsive. The gateway might be down or the cache might be unreachable.

- Verify that the proxy ports specified in `config.json` are not used by another service.
- If the DataPower Gateway is in a container, verify that the ports specified in `v10-cache-proxy-port`, `v5-cache-proxy-port`, or both, are exposed in the container. The exposed container ports must be specified in `v10-cache-container-port`, `v5-cache-container-port`, or both.
- Verify that the correct host interface (IP address) is selected for the cache proxy at runtime.
- If the v5 QES is unresponsive, check the following settings.
  1. Verify that the v5 QES uses the default address and that it is enabled.
  2. Check the logs in debug mode and search for a line that contains **Making TCP Proxy with URL**. If the log contains this line, check for an error that explains why the TCP proxy was not created. If the error is present, fix the error to allow the proxy to connect to the QES, then run the TMU again.
  3. If the log contains the line **Making TCP Proxy with URL** but there is no error, verify that there is a TCP proxy service configuration called `TMUProxyForMigration`. The remote IP address and remote port must be set to the QES default address and port. If necessary, modify the proxy configuration with correct information to connect to the QES, and then run the TMU again. If `TMUProxyForMigration` does not exist, create it and configure it to connect to the QES, and then run the TMU again.
- If the v10 gateway-peering token cache is unresponsive, check the following configuration settings.
  1. Verify that the gateway-peering object is enabled and specified in `config.json`.
  2. Check the logs in debug mode and search for a line that contains **Gateway peering info**. Verify that the information for the gateway-peering object is correct as specified in `config.json`.
  3. If TLS is disabled, check the logs in debug mode and search for a line that contains **Making TCP Proxy with URL**. If the log contains this line, check for an error that explains why the TCP proxy was not created. If this error is present, fix the error and then run the TMU again.
  4. If the log contains the line **Making TCP Proxy with URL** but there is no error, verify that there is a TCP proxy configuration called `TMUProxyForMigration` in DataPower Gateway. The remote host and port must be set to the local host and port of the gateway-peering object. If necessary, modify the proxy with the correct information to connect to the gateway-peering object, and then run the TMU again. If `TMUProxyForMigration` does not exist, create it and configure it to connect to the gateway-peering object, and then run the TMU again.
  5. If TLS is enabled, check the logs in debug mode and search for lines that contain **Making TLS Client Profile with URL** and **Making TLS Proxy with URL**. If the log contains these lines, check for an error that explains why the TLS client and TLS proxy were not created. Fix the error and run the TMU again.
  6. If the log contains the lines **Making TLS Client Profile with URL** and **Making TLS Proxy with URL** but there is no error, verify that there is a TLS proxy configuration called `TMUProxyForMigration` and a TLS client profile configuration called `TMUProfileForMigration` in DataPower Gateway. The remote host and port of the proxy must be set to the local host and port of the gateway-peering object. If necessary, modify the configurations to connect to the gateway-peering token cache, and then run TMU again. If either the TLS proxy called `TMUProxyForMigration` or the TLS client profile called `TMUProfileForMigration` do not exist, create these objects, configure them to connect to the gateway-peering object, and then run the TMU again.

Failed to establish a connection with {address:port}.

The gateway may be down or the DataPower Gateway REST management interface may be down.

- Verify that the gateway and the REST management interface are up.
- Verify that the port in the message is the port of the REST management interface. If it is not, modify the `v5-rest-management-interface-port` or `v10-rest-management-interface-port` field in `config.json`.
- If the DataPower Gateway is in a container, verify that v5 and v10 REST management interface ports are exposed by the container that is mapped to the DataPower Gateway REST management interface.

Could not get Gateway Peering information.

The DataPower Gateway REST management interface could not retrieve gateway-peering information.

- Verify that the gateway-peering object specified in `config.json` is created and enabled in the chosen DataPower Gateway domain.

## Migrating custom policies

The migration utility migrates API Connect Version 5 user-defined custom policies to API Connect Version 10. After migration, the v5 custom policies run on a v5-compatible DataPower Gateway or the DataPower API Gateway. Custom policies can be migrated to API Gateway with limitations.

## Overview of the custom policy file format

Review your custom policies in the `/ccloud` hierarchy that is created by the migration upack. The path to custom policies in the `/ccloud` hierarchy structure is:

```
provider-orgs -> <provider-org-name> -> catalogs -> <catalog-name> -> configured-gateway-services -> <gateway-name> -> policies
```

The following example from a `/ccloud` hierarchy shows custom policies for the catalog `policy-and-vendor-ext-test` listed under the gateway service `g-w-1-8f82`:

```
./provider-orgs/onlinebanking/catalogs/policy-and-vendor-ext-test/configured-gateway-services/
└─ g-w-1-8f82
 └─ configured-gateway-service.yml
 └─ policies
 ├── client-i-p-filter-policy-9d58-1.0.doc.yml
 ├── client-i-p-filter-policy-9d58-1.0.meta.yml
 ├── client-i-p-filter-policy-9d58-1.0.zip
 ├── client-i-p-filter-policy-9d58-2.0.doc.yml
 ├── client-i-p-filter-policy-9d58-2.0.meta.yml
 ├── client-i-p-filter-policy-9d58-2.0.zip
 ├── validate-auto-xsd-jsv-1.0.0.doc.yml
 ├── validate-auto-xsd-jsv-1.0.0.meta.yml
 ├── validate-auto-xsd-jsv-1.0.0.zip
 ├── validate-username-token-5.0.0.doc.yml
 ├── validate-username-token-5.0.0.meta.yml
 └── validate-username-token-5.0.0.zip
```

File naming legend:

- `*.doc.yml` - the policy definition files
- `*.meta.yml` - metadata used only by the migration utility.
- `*.zip` - the custom policy zip files

## Migrating custom policies to a v5-compatible gateway

A Version 5 custom policy definition is a zip file containing the yaml description of the policy (its name and configuration parameters) and the implementation zip of DataPower configuration that runs on the gateway. The zip file is pushed onto the gateway service used by any catalog that has APIs that make use of the policy. For more information on v5 custom policies, see [User-defined policies in API Connect](#).

The migration utility migrates both the zip file and the yaml description file. After migration, the v5 custom policies run on a v5-compatible DataPower Gateway. Note that the structure of the custom policy zip file is the same in v5-compatible gateway as in v5. The v5-compatible gateway runtime will run any custom policy that a v5 gateway runs.

## Migrating custom policies to API Gateway

Important: When migrating custom policies to v10 API Gateway, the v5-emulation layer is used to ease compatibility. For more information, see [v5 policy emulation limitations](#).

You can migrate v5 custom policies to API Gateway using the v5-emulation layer. The migration utility bundles custom policies and groups them by gateway service during the port step. During the push step, the custom policies are pushed to the Gateway Service with instructions for DataPower to enable the v5-emulation layer, along with any v5-emulated or wrapped policies that were specified during the port step, such as `invoke_1.5.0` or `gatewayscript_1.0.0`.

v5 policy emulation is deployed by the API Connect Gateway Service using a Gateway extensions manifest. For more information, see [Gateway extensions manifest](#).

Note: If porting to API Gateway using the `custom-policies-scope=catalog` parameter, and spaces are contained in the v5 backup, then any duplicate custom policies found will be discarded. The final remaining custom policy out of the duplicates when pushed using `archive:push` will be pushed to all spaces within the same catalog. This is the correct behavior of the v10 API migration utility. For more details about pushing custom policies at the catalog (space) scope, see [Importing a user-defined policy into a Catalog](#).

## Migrating gateway extensions

Use the appropriate procedure for migrating extensions to either a v10 DataPower® Gateway (v5 compatible) or a v10 DataPower API Gateway.

## Migrating gateway extensions to a v10 DataPower Gateway (v5 compatible)

### About this task

Follow these steps to migrate gateway extensions to a v10 DataPower Gateway (v5 compatible).

### Procedure

Follow all migration utility steps to migrate to a v5-compatible DataPower Gateway. For instructions, see [Migration steps](#).

The gateway extensions are automatically extracted as part of the `archive:unpack` step. No further manual steps are required.

### Results

The v5 extension zip file is pushed to the DataPower Gateway (v5 compatible) service.

Note: Only one gateway extension per gateway service can be used.

# Migrating gateway extensions to a v10 DataPower API Gateway

## Before you begin

Note: When migrating v5 extensions to v10 API Gateway, the v5-emulation layer is used to ease compatibility for any pre-request or post-response global assemblies it contains. For more information, see [v5 policy emulation limitations](#).

The following limitations apply to migrating gateway extensions to a v10 DataPower API Gateway.

- Post-request assemblies are not compatible with the API Gateway and cannot be migrated using this process. Pre-request, post-response, and post-error can be migrated using these steps.
- Any modifications to the DataPower v5 Multiprotocol Gateway cannot be migrated using these steps because the API Gateway does not use a Multiprotocol Gateway. The modifications will still be in the `gateway-extension.zip` that is pushed to API Manager, and the artifacts might still be imported to DataPower. These artifacts can be deleted from the DataPower appliance.
- Any load-balancer modifications cannot be migrated using these steps as the load-balancer DataPower object is not used in the same manner in the API Gateway. These artifacts can be deleted from the DataPower appliance.

## About this task

You can migrate pre-request and post-response global assemblies contained within v5 extensions to API Gateway using the v5-emulation layer. The migration utility bundles the entire v5 extension file and groups them by gateway service during the `port` step. During the `push` step, the v5 extensions are pushed to the Gateway Service, and pre-hook and post-hook global assemblies are enabled using global policy files that were generated previously during the `port-to-apigw` step. DataPower object modifications or additions that are not v5 framework changes or load-balancer changes are also automatically imported during this step.

Global assemblies that are pre-request or post-response are utilized through the v5-emulation layer as an Assembly Function. They are referenced by the global policy files auto-generated by the migration utility during the `port-to-apigw` step.

## Results

The `gateway-extension.zip` file is pushed to the DataPower API Gateway service.

Note: Only one gateway extension per gateway service can be used at a time. However, the manifest extension allows a vector of multiple extensions contained within the single `gateway-extension.zip` that is pushed to the single gateway service.

## Migration considerations

You can optimize the migration process by reviewing the migration considerations and taking any actions that apply to your deployment.

Review the following information:

- [Limitations and workarounds](#)
- [Version 5 data that is not migrated](#)
- [Parity between v5 and v10](#)
- [Customizations to recreate](#)

## Limitations and workarounds

- When migrating catalog spaces in a provider organization that uses a local user registry (LUR), ensure that the LUR is configured with the visibility setting of either Public or Custom. If set to Custom, ensure that the provider organization is selected.
- Migration of the v5 username token policy for authenticating the username token is not supported.
- In v10, you cannot publish a product containing an API with a `regex` Swagger property that includes the ( or ? in the characters for the pattern. You can expect to see an error message similar to the following example:

```
(? The preceding token is not quantifiable
```

- The migration utility validates Products and APIs against the OpenAPI specification (validation is not performed on draft Products and draft APIs). The `archive:unpack` command issues warnings for any Product or API that fails validation. To resolve warnings, fix the Product or API and re-run the `unpack` command. Example warning:

```
WARN[0019] Error validating product cloud/provider-orgs/onlinebanking/catalogs/test/products/clientipfilter-policy-test-api-1.0.0.yml:
Validation errors:
Error validating API cloud/provider-orgs/onlinebanking/catalogs/test/products/api123-1.0.0.api.yml
- Must validate one and only one schema (oneOf) (context: (root).securityDefinitions, line: 36, col: 3)
- securityDefinitions.authorizationUrl Does not match format 'uri' (context: (root).securityDefinitions.authorizationUrl, line: 0, col: 0)
```

- OAuth Provider APIs:
  - To ensure that APIs that use OAuth security definitions are mapped to the corresponding OAuth Provider APIs in v10, make sure that either the Authorization URL or the Token URL defined in OAuth Security Definitions of APIs that use OAuth correctly point to the base path of the OAuth Provider API. Note that the Migration tool uses that logic to map Consumer APIs to Provider APIs, and adds `x-ibm-oauth-provider` in the OAuth Security Definition of the consumer API.
  - To verify whether OAuth Provider APIs are migrated correctly, ensure that all the APIs that use OAuth Security Definition have the new field `x-ibm-oauth-provider` added. Check the value of that field and make sure it points to a correct OAuth Provider resource object that was created and configured in the catalog to which the API is published.
  - In v5, OAuth Providers are a specialized type of API that are published within products. In v10, OAuth Providers are first class objects that are directly published in the provider organization and are not APIs that are published within products. Because of this, it is possible that you may have some v5 products that only contained OAuth Provider APIs. Since these products do not contain any Open API definitions in v10, they will not be published as all products in v10 must contain at least one Open API definition.
- Portal Delegated User Registry migration limitations
  - OIDC: Automated migration is not supported for some v5 PDUR OIDC registries and you will see instructions for providing the `client_id` and `client_secret` while mapping the registries.

- LDAP: Automated configuration is not supported for LDAP registries and you will see instructions for configuring them manually while mapping the registries.
- The SCIM (System for Cross-domain Identity Management) user registry type was supported in Version 4 but was deprecated for Version 5 and is not supported in v10. If data with a SCIM identity provider type is detected during migration of v5 data, the migration utility logs the following error message:

```
WARN[0614] manual fix required for cloud/provider-orgs/apimteam/user-registries/scimcv/user-registry.yml :
The SCIM user registry type is no longer supported
```

Any v5 data with a SCIM identity provider type must be manually migrated. For v10, you can use OIDC with a back-end user registry that supports SCIM. You must develop custom code to complete the data flow between the OIDC provider and your backend user registry. The API Connect built-in OIDC registry interacts with your OIDC provider.

- Roles limitations
 

In API Connect v5, it was possible to create a role (and associated permissions) at the organization level and the role was inherited by the catalogs within the organization. With the v5 inheritance model it was possible to modify the inherited role and add or remove permissions.

In API Connect v10 the inheritance model has changed and it is no longer possible to remove permissions from an inherited role. Because of this design change, it becomes difficult to perform a true 1-to-1 migration keeping the original parent-child relationships. For example, in v5 it was possible for an inherited role to contain fewer permissions than its parent role, but this is not supported in v10. To work around this restriction, the parent-child relationship is purposely decoupled during migration, and a new artificial parent role is created for each of the child roles.
- Modifications made to default roles in v5 are not migrated to v10.
- If the v5 settings `server_name_indication` and `limit_renegotiation` are not specified, the value defaults to `true`. However if they are specified with value of `false` that value is changed to `true`. Thus the settings are always `true`.
- Subscriptions on staged products are not automatically migrated because in v10, staged products cannot have subscriptions on them. To save those subscriptions, change the state of the product to either deprecated or published before running the `archive:push` command to upload data to v10. You can do this either in v5 before downloading your data, or in by mapping or editing the downloaded product metadata and to change the state of the product. If you have already pushed the staged product, then you can publish the product in your v10 environment and rerun `archive:push` to migrate the subscriptions.
- The migration utility appends `-apigw` to the end of the `x-ibm-name` of APIs to maintain uniqueness from the original v5 artifacts. Therefore, custom backend services that rely on the `x-ibm-name` of the APIs to be matched must be updated to look for `-apigw` in the new API name. Alternatively, the custom backend service can be modified to strip `-apigw` from the API name.
- When migrating default catalog user registries, and migration has completed, you might see more than one catalog User Registry with the same title. Whenever a new catalog is created in v10, API Connect creates a default catalog user registry. When migration takes place, the catalog user registry from v5 is migrated onto the v10 system. Both these registries receive the same Title; however they receive different Names and function as separate user registries (registries are managed by Name, not Title).

Example: If the catalog is called `test`:

- Name of user registry migrated from v5: `test-catalog`
- Name of default user registry created in v10: `test-catalog-0`
- Title of both the user registries: `Test Catalog User Registry`

All the user data is successfully migrated to the user registry coming from v5; manually update the catalog settings to use the migrated v5 user registry as the default. You can optionally clear the default user registry provided by v10 from the list of configured catalog user registries.

If you manually create the catalog and users already exist in the default catalog user registry provided by v10, then both the migrated v5 catalog user registry and the v10 default user registry can coexist, and either of them can be selected as default.

## Version 5 data that is not migrated

- APIs with LTPA policies are not migrated because this policy is not supported in v10.
- Configuration in v5 for gateway type `Microgateway` is not migrated because `Microgateway` is not supported in v10.
- Transient or conflicting configuration data is not migrated:
  - Tasks that are still active are not included in the downloaded v5 data; for example, lifecycle approvals or subscription approvals that have been requested but not granted.
  - Drafts on v5 that are local are not migrated. If you have draft APIs or draft products that you want migrated to v10, use `apic:drafts:push` to upload in v5 before you download your v5 data for migration. For more information, see [Working with drafts](#) in the v5 Public Cloud documentation.
  - Pending invitees in user registries are not migrated.
  - Pending invitations for ownership of provider organizations are not migrated; however consumer organizations with pending ownership are migrated. If the v5 provider organization owner received an invitation for ownership but never accepted it, the owner identity is not migrated, which causes the organization to not be migrated. To avoid this issue, check for any unaccepted invitations prior to migration, and ensure they are accepted.
  - Conflicting user names within the same identity provider are not migrated.
  - Conflicting OAuth provider names or base paths are not migrated. If the migration utility encounters conflicting names or base paths for OAuth providers, the provider is not migrated.
- OAuth Shared secret keys are not migrated unless you explicitly [map them in the mapping.yml file for the gateway](#).
- Organizations without an active owner (user identity) are not migrated. One effect of this is that if the original owner of the organization is an identity that was not migrated, the provider organization or consumer organization cannot be migrated. Prevent this issue by verifying that the owner of the organization is a current user.
- Analytics data is not migrated. Analytics data (including searches, visualizations, and dashboards) cannot be migrated due to changes in the Analytics feature. If you want to preserve your event data, you should export it from v5 Public Cloud using either [CSV files](#) or [REST API calls](#) as explained in the v5 Public Cloud documentation.

## Parity between v5 and v10

- The migration utility sluggifies (a slug is a human-readable, unique identifier) the names and versions of some elements when migrating from v5 to v10 to ensure that each element in v10 passes Open API schema validation and is uniquely named. The conversion rules differ depending upon which version you are migrating to.
  - **Version 10.0.5.1 and earlier:** For the names of API properties, Catalogs, Provider Organizations, Consumer Organizations, Spaces, and some other elements, any character other than `a-z`, `0-9`, or `-`, is converted to `-` and a unique suffix (based on the checksum of the name or version) is appended. Uppercase letters are changed to lowercase, with `-` added between words. Leading `-` characters are removed, and multiple consecutive `-` characters are reduced to a single `-`. For versions of similar elements, the `.` (period) character is also allowed.



- **Version 10.0.5.2 and later:** For the names and versions of API properties, Products, Consumer Organizations, Space and some other elements, any character other than **a-z, A-Z, 0-9, -, \_,** or period (**.**), is converted to **-** and a unique suffix (based on the checksum of the name or version) is appended. Catalog and Provider Organization names follow the same rules, except that they do not support the period (**.**) character. Names or versions with uppercase characters are not converted to lowercase, but a unique suffix is appended to them.

If you are using the latest version 10.0.5.2 or later version of the migration utility, you can use the `--use-old-sluggify` parameter with the `apicm archive:unpack` command to unpack the V5 configuration data onto the local filesystem using the version 10.0.5.1 conversion rules. You can use this option in the following scenarios.

- You are starting a new migration to API Connect v10.0.5.1 with version 10.0.5.2-R4 or later of the migration utility.
- You started the migration using version 10.0.5.1 of the migration utility and have upgraded to version 10.0.5.2-R4 or later.

For more information, see the migration instructions that apply to your deployment.

- [Migration steps with PDUR](#)
- [Migration steps without PDUR](#)

Table 1. Example migrated names on v10

| v5 name  | Migrated name on v10.0.5.1 or earlier                            | Migrated name on v10.0.5.2 or later                              |
|----------|------------------------------------------------------------------|------------------------------------------------------------------|
| abc-def  | abc-def (no change)                                              | abc-def (no change)                                              |
| Abc      | abc-a2c6                                                         | Abc-a2c6                                                         |
| AbcDef   | abc-def-dfc                                                      | AbcDef-dfc                                                       |
| Abc Def  | abc-def-1e82<br>With <code>sluggify-no-checksum</code> : abc-def | Abc-Def-1e82<br>With <code>sluggify-no-checksum</code> : Abc-Def |
| abc_def  | abc-def-c2bb                                                     | abc_def (no change)                                              |
| abc--def | abc-def-813d                                                     | abc--def (no change)                                             |
| -abc-def | abc-def-abb                                                      | -abc-def (no change)                                             |
| ABC-DEF  | abc-def-d1e                                                      | ABC-DEF-d1e                                                      |

Table 2. Example migrated versions on v10

| v5 version | Migrated version on v10.0.5.1 or earlier | Migrated version on v10.0.5.2 or later |
|------------|------------------------------------------|----------------------------------------|
| 1.0.0      | 1.0.0 (no change)                        | 1.0.0 (no change)                      |
| 1.0_2      | 1.0-2-3933                               | 1.0_2 (no change)                      |
| 1.0 2      | 1.0-2-913                                | 1.0-2-913                              |
| 1.0--2     | 1.0-2-5410                               | 1.0--2 (no change)                     |
| -1.0-2     | 1.0-2-26cf                               | -1.0-2 (no change)                     |

You can use `apicm sluggify <string>` to view how a name or version string will be sluggified. The following examples are for version v10.0.5.2 or later:

```
$./apicm sluggify Abc
Abc-a2c6
$./apicm sluggify -v 1.0_2
1.0_2
$./apicm sluggify --version '1.0 2'
1.0-2-913
```

- In v10, Spaces are not supported in the Sandbox Catalog. Spaces in Sandbox Catalogs are not migrated from v5 to v10. The names of Spaces found in v5 Sandbox Catalogs are logged in the migration output log. To preserve the Spaces, use a mapping file to rename the Sandbox Catalogs.
- In v10, publishing an API fails if `enforced` is set to `true`, and `phase` is set to either `Specified` or `Identified` instead of `Realized`. During migration, the `phase` will be set to `Realized` to allow the API to be published.
- Enforcement of user subscription to an API is handled differently in v5 and v10. In v5, subscription to a product is not allowed when in the following scenario: An API is created with `enforced` set to `false`, the API is associated with a product, the product is published, and then a user attempts to subscribe to a Product or Plan from the Developer Portal. In v10, the subscription is allowed for this scenario. This change can affect migration because a v5 Product that could not be subscribed to in v5 can be subscribed to in v10. Note that the change in behavior for v10 accommodates the scenario where Products or Plans have multiple APIs, some of which are enforced and some of which are not enforced. The new behavior allows such Products or Plans to be subscribed to.
- In v10, the API Manager handles all authentication methods. This change eliminates the need for configuration of a catalog with a Portal Delegated User Registry (PDUR), so during migration you will map PDURs to v10 standard user registries (see [Migration steps with PDUR](#).)
- If `OpenAPI.info.contact.email` contains more than one email address, only the first email address is migrated. If the email is not in valid RFC5322 email format then it is removed during migration.
- Provider-side applications:  
In API Connect, registered applications are a fundamental concept representing an application identity known to the system. Applications have their own credential sets in the form of client ID and client secret pairs, and access to published API endpoints is governed by subscriptions associated with registered applications. Most commonly, registered applications are created by app developers belonging to a consumer organization in the context of a developer portal.

However, API Connect v5 also provides the ability for provider organizations to register and manage applications, independent of any consumer organization. Provider-owned apps behave in the same way as consumer-owned apps and carry credentials and subscriptions in exactly the same way. Customers use provider-owned applications for API testing or for cases where the app and API are owned and managed directly by the same team.

The API Connect v10 platform API does not support the creation of provider-side apps, so migration of provider-side apps is only possible if they are converted to consumer-side apps. Each provider-side app is migrated as follows by the API Connect migration utility:

- An instance of the app including its credentials (sets of client ID and secret) is created in each catalog belonging to the provider organization.
- The app is owned by a consumer organization which is the existing Sandbox Test Organization (for the provider organization's automatically created sandbox catalog) or a consumer organization named `Catalog-name` Test Organization (for other catalogs).
- The consumer organization's owner is a user named `test-user`. This user name already exists in the context of a provider organization's automatically created sandbox catalog.
- The consumer organization's owning user belongs to a user registry named `catalog-name-test-org`.
- The subscriptions belonging to the original provider-side app are retained.

These definitions are added to the relevant locations subordinate to the `cloud/` directory that is produced by the `apicm archive:unpack` command. When an `apicm archive:push` is performed, the definitions are propagated to the target system as part of the overall configuration.

## Customizations to recreate

Version 5 provides methods to customize some configurations. After you complete the automated migration, manually reapply the following customizations.

- Customized email templates  
Customized email templates are not migrated from Version 5. You can customize email templates in Version 10 with greater flexibility and precision. See [Customizing email templates](#).
- Developer Portal customizations are not migrated to a new Drupal version  
For the Developer Portal, none of the customizations are migrated because the v10 Drupal-9 baseline differs greatly from the v5 Drupal-7 baseline. This means that custom portal themes, custom modules, custom fields, custom blocks, custom pages, and more cannot be migrated. For more information, see [Migrating your Developer Portal from Version 5 to Version 10](#).

---

## Migration troubleshooting

If you run migration and observe error messages in your log output, you can take troubleshooting steps, and run the migration again.

Refer to the following sections for information on resolving migration issues:

- [apicm command line help](#)
- [Log files for apicm](#)
- [Common errors](#)
- [Error messages and warnings from archive:unpack](#)
- [Error messages and warnings from archive:push](#)
- [Troubleshooting tips](#)
- [Failure to convert API definitions to API Gateway](#)
- [Custom policies not appearing in the policy palette after migration](#)

---

## apicm command line help

You can use command line help to review usage syntax for `apicm`. For example:

```
./apicm --help
API Connect Migration Utility (AMU)
```

```
Usage:
 apicm [command]
```

```
Available Commands:
 archive:apistatus Generate stats about API use
 archive:automation Generate summary of the data
 archive:gstats Generate stats about gatewayscript cmd use
 archive:port-to-apigw Port artifacts to apigw
 archive:push Push unpacked content to APIConnect server
 archive:unpack Unpack a V5 management/portal archive
 help Help about any command
 login Log in to IBM API Connect
 sluggify Sluggify a name or version string
 version APIConnect migration toolkit version
```

```
Flags:
 --accept-license Accept the license for API Connect
 -h, --help Help for apicm
```

Use "`apicm [command] --help`" for more information about a command.

Example usage for more information about a command:

```
$./apicm archive:push --help
Push unpacked content to APIConnect server
```

```
Usage:
 apicm archive:push CLOUD_MANAGER_ENDPOINT/API_MANAGER_ENDPOINT ARTIFACT_PATH
```

Locate `CLOUD_MANAGER_ENDPOINT` and `API_MANAGER_ENDPOINT` in the `apiconnect-up.yml` file used during `apic install` [flags]

```
Flags:
 --apigw-only Only attempt to push APIGW artifacts
 --catalog strings Provide a comma separated list of catalogs to push (e.g. --catalogs=cat1,cat2)
 --dry-run Prepare the working directory but do not push to server
 -h, --help help for archive:push
 --no-drafts No draft APIs or Products will be pushed to the server
 -f, --overwrite Overwrite draft APIs and products, published APIs and products,
 and OAuth providers if they exist on the server
 --saveDebugLog string[="default"] Debug logs are saved by default to logs/debug/archive-push<timestamp>debug.log
 --saveErrorLog string[="default"] This flag saves the file with a custom name (e.g. --saveDebugLog=debugLog.txt)
 --saveInfoLog string[="default"] Error logs are saved by default to logs/archive-push<timestamp>error.log/
 --saveInfoLog string[="default"] This flag saves the file with a custom name (e.g. --saveErrorLog=myLog.txt)
 --saveInfoLog string[="default"] Info logs are saved by default to logs/archive-push<timestamp>info.log
 --saveInfoLog string[="default"] This flag saves the file with a custom name (e.g. --saveInfoLog=myLog.txt)
```

```
Global flags:
 --accept-license Accept the license for API Connect
```

---

## Log files for apicm

The `apicm` command creates log files that you can use to review messages produced by the `unpack` and `push` actions. All logs are saved in the `logs` directory. Unique files are saved for each run. All default log names can be overridden via the command line.

#### Error log

Includes all warning and error level messages.  
Error logs are saved by default to:

- `logs/archive-unpackerror.log/`
- `logs/archive-pusherror.log/`
- `logs/archive-port-to-apigwerror.log/`

You can optionally give the file a different name:

```
apicm --saveErrorLog= '<filename>'
```

For example:

```
apicm archive:unpack apic33-2020-02-19-1650.tar.gz --saveErrorLog= 'unpack.txt'
```

This command creates a file `unpack.txt` in the logs directory:

```
~/example/migration-cli/logs/unpack.txt
```

#### Info Log

Includes all info, warning and error level messages.  
Info logs are saved by default to:

- `logs/archive-unpackinfo.log/`
- `logs/archive-pushinfo.log/`
- `logs/archive-port-to-apigwinfo.log/`

You can optionally give the file a different name:

```
apicm --saveInfoLog= '<filename>'
```

For example:

```
apicm archive:unpack apic33-2020-02-19-1650.tar.gz --saveInfoLog= 'unpack_info.txt'
```

This will create a file `unpack_info.txt` in the logs directory:

```
~/example/migration-cli/logs/unpack_info.txt
```

Default path for info logs?

#### Debug Log

Includes all info, warning, error and debug level messages. If you encounter errors or limitations that are not documented on this page, access the debug log.  
Debug logs are saved by default to:

- `logs/debug/archive-unpackdebug.log`
- `logs/debug/archive-pushdebug.log`
- `logs/debug/archive-port-to-apigwdebug.log`

Contact your IBM representative for help with using the Debug log.

## Common errors

---

You can use the `--saveErrorLog` flag to specify that all the errors are saved in a file in `logs` folder, as described in [Saving the error log](#). Common errors:

- Long chain of error messages hours into a `push`:  
This scenario typically occurs due to expiration of an authentication token. To resolve this issue, either reauthenticate and or extend the token timeout. See [Token expiration](#).
- Unable to find User Registry when migrating (pushing) a provider-org:  
Occurs when the `admin-org` user registry artifacts are not shared prior to pushing provider organization artifacts. To resolve the issue, use the cloud admin account to change the visibility of the registry to `public`, and `push` the provider organization again.
- Unable to find `xxxx` when migrating (pushing) a provider organization:  
Occurs when `admin-org` artifacts are not pushed prior to pushing provider organization artifacts. Ensure that all of admin org data is pushed without errors, and then `push` your provider organization again.
- Unable to create an identity provider portal-delegated-identity-provider `xxxx` because the PDUR backup data is not provided.  
This error occurs if certain Portal delegated user registries are missing from the PDUR extract provided. If the missing user registries are not no longer used, they can be skipped. If they are still used, you must create a fresh backup of the PDUR.
- Unable to create product `xxxx`. The OpenAPI definition `<xyz>` requires the policy `<sample-policy>` that is not configured on the gateway-service `<sample-gateway>`:  
This error occurs when the specified policy is not configured on the gateway for the catalog that contains the product. To resolve this, you must configure, on the v5 system, the specified policy on the gateway for the catalog, and re-extract the data.
- Fixing API Errors  
Example:

```
WARN[2002-01-10T15:06:16-05:00] AMUUAU005W: product oauth-apis:1.0.0 requires updating before it can be pushed to a target system
Reason: product oauth-apis:1.0.0 includes content with validation errors:
Error validating API cloud/provider-orgs/onlinebanking/catalogs/oauthcatalog/products/api050-1.0.0.api.yml
- Must validate one and only one schema (oneOf) (context: (root).securityDefinitions, line: 35, col: 3)
```

```
- securityDefinitions.authorizationUrl Does not match format 'url' (context:
(root).securityDefinitions.authorizationUrl), line: 0, col:0)
```

For example, where the issue is in the API:

```
securityDefinitions:
 ClientID:
 in: query
 name: client_id
 type: apiKey
 OAuthDef:
 authorizationUrl: ""
 description: ""
```

- v5 emulated policies and custom policies not deployed to gateway. Make sure the following was done during the migration steps:
  - xml-manager is enabled in default domain of all gateways
  - Mapping of gateways was done before the porting step
  - gateway-extension.zip shows up under cloud/admin-org/availability-zones/availability-zone-default/gateway-services/<gw-name>/gateway-extension.zip
  - Ensure push of admin-org is done after all of the above is verified
- Missing Subscriptions  
One of the causes for missing subscriptions could be that the products associated with those subscriptions have not been created successfully. Make sure to check the logs and see if there have been any errors during creation of those products. Please also note in these cases there would be no logs related to those missing subscriptions.

## Error messages and warnings from archive:unpack

---

"AMUUAU000E": "The migration tool failed to initialize."

The migration utility could not access files or directories needed to establish its working environment. Ensure that file permissions allow write access to the current working directory and that the cloud directory is empty or non-existent.

"AMUUAU001E": "The {{.Type}} {{.Name}} contains content that could not be unpacked and will not be included."

The migration utility is unable to read the API Connect v5 representation of an object in the migration archive. This content is skipped and will not appear in the unpacked migration data.

"AMUUAU002F": "The archive content at {{.DocPath}} cannot be read."

The migration utility is unable to read the internal contents of a collection in the API Connect v5 migration archive. This is likely to indicate a corrupt migration archive file. The archive:unpack operation cannot be completed.

"AMUUAU003E": "The portal delegated user registry file {{.File}} cannot be unpacked and will be skipped."

The migration utility is unable to read a PDUR archive. User information associated with this user registry will not be included in the unpacked migration contents.

"AMUUAU004E": "The archive content at {{.FilePath}} cannot be read and will not be unpacked."

The migration utility is unable to read the internal contents of an individual file in the API Connect v5 migration archive. This is likely to indicate a corrupt migration archive file. This content is skipped and will not appear in the unpacked migration data.

"AMUUAU005W": "{{.Type}} {{.Name}} requires updating before it can be pushed to a target system."

The migration utility has unpacked a configuration object for which modifications are required because the content is incorrect or missing information required by a target API Connect v10 system. The unpacked file must be updated by the user before it can be published successfully to the target system.

"AMUUAU006W": "The {{.Type}} {{.Name}} has been linked to OAuth provider {{.Provider}}. Please update the security definition {{.Security}} if another OAuth Provider should be used instead."

The migration utility has added a reference from an OAuth security definition in the named API to an OAuth provider resource. The migration utility uses the authorization URL, token URL and scope information where possible to determine which OAuth provider resource is to be used with each OAuth-secured API. In some cases, the migration utility cannot accurately determine which OAuth provider is intended for use with a particular API. The OAuth provider reference can be modified if needed by editing the security definition in the unpacked API file and changing the x-ibm-oauth-provider property value to match the intended OAuth provider resource. The reference value is a relative file path that identifies an unpacked OAuth provider resource.

"AMUUAU007W": "The {{.Type}} {{.NameVersion}} could not be linked to an OAuth provider. Please update the security definition {{.Security}} to select an OAuth provider."

The migration utility has been unable to add a reference from an OAuth security definition in the named API to an OAuth provider resource.

The migration utility uses the authorization URL, token URL and scope information where possible to determine which OAuth provider resource is to be used with each OAuth-secured API. In some cases, the migration utility cannot accurately determine which OAuth provider is intended for use with a particular API. The OAuth provider reference can be added by editing the security definition in the unpacked API file and setting the x-ibm-oauth-provider property value to match the intended OAuth provider resource. The reference value is a relative file path that identifies an unpacked OAuth provider resource.

"AMUUAU008W": "{{.Type}} {{.Name}} could not be introspected for OAuth provider API references."

The migration utility has attempted to determine which OAuth providers should be configured in support of an API product, but a reference to an OAuth provider API in the product cannot be correlated with an OAuth provider resource.

"AMUUAU009W": "{{.Type}} {{.Name}} contains invalid content that cannot be unpacked."

The migration utility has encountered an entry in the API Connect v5 migration archive that contains unexpected or corrupt data. This content is skipped and will not appear in the unpacked migration data.

"AMUUAU010E": "{{.Type}} {{.Name}} cannot be unpacked to the local filesystem."

The migration utility has attempted to unpack an entry in the API Connect v5 migration archive that could not be written to the local filesystem during the archive:unpack operation. This may be the result of a local file permissions problem, an unexpectedly long file name resulting from the unpack, or a shortage of space on the local filesystem.

"AMUUAU011E": "{{.Type}} {{.Name}} contains a {{.Target}} reference that cannot be resolved."

The migration utility has encountered a SOAP API entry in the API Connect v5 archive that refers to a WSDL definition that cannot be found in the archive. This entry will not be included in the unpacked content.

"AMUUAU012W": "The {{.Type}} {{.Name}} contains a {{.Target}} reference that cannot be resolved."

The migration utility has encountered an entry in the API Connect v5 archive that refers to another entry that cannot be found in the archive. This entry will not be included in the unpacked content.

"AMUUAU013E": "Portal delegated user registry information could not be loaded for user registry {{.Name}}.  
The migration utility has encountered a portal delegated user registry definition in the API Connect v5 archive for which no PDUR file name has been provided as an argument to the archive:unpack command. The user registry contents will not be included in the unpacked content."

"AMUUAU014E": "The migration tool failed to load content for the admin organization."  
The migration utility was unable to find the admin organization while unpacking provider organization information in the API Connect v5 archive. This is likely to result in further errors while unpacking the provider organization contents."

"AMUUAU015E": "The migration tool failed to load provider organization contents."  
The migration utility was unable to find the load provider organization contents while unpacking the API Connect v5 archive. This message indicates an internal error with the migration utility."

"AMUUAU016E": "The {{.Type}} information belonging to organization {{.Org}} could not be unpacked."  
The migration utility was unable to process the provider organization contents while unpacking the API Connect v5 archive. This message indicates an internal error with the migration utility."

"AMUUAU017W": "The settings update for {{.Type}} {{.Name}} could not be completed."  
The migration utility was unable to update the settings for a configuration object while unpacking the API Connect v5 archive."

"AMUUAU018E": "Apps belonging to provider organization {{.Org}} could not be converted to consumer apps."  
The migration utility was unable to convert registered applications belonging to the provider organization into applications in each catalog. The conversion occurs because the representation of applications differs between API Connect v5 and API Connect v10. This message indicates an internal error in the migration utility."

"AMUUAU019W": "User registry {{.UserRegistryName}} must be migrated to a different user registry type."  
The migration utility has encountered a user registry in the API Connect v5 migration archive that is not directly supported in API Connect v10. User entries can be migrated if a mapping file is used to override the user registry name and refer to another user registry that has been created in the target API Connect v10 environment."

"AMUUAU020W": "User {{.User}} in portal delegated user registry {{.Registry}} is not associated with any OIDC provider and will be skipped."  
The migration utility has encountered a user entry in the API Connect v5 migration archive that belongs to a portal delegated user registry but cannot be associated with any backing identity provider. This can occur if no PDUR export file has been provided, or if the user entry information is incomplete."

"AMUUAU021W": "The TLS profile {{.Name}} includes an unsupported protocol {{.Protocol}} that will not be included in the configuration."  
The migration utility has encountered a TLS profile entry in the API Connect v5 migration archive that uses a protocol that is not supported in the API Connect v10 environment. This protocol will be omitted from the unpacked entry for the TLS profile."

"AMUUAU022W": "User {{.UserName}} ({{.ID}}) will need a password reset"  
The migration utility has encountered a user entry in the API Connect v5 migration archive that is associated with a local user registry but does not have a password configured. The user entry will be migrated, but the user's password reset must be reset before logging in. This may occur when user registry mappings are used to migrate users to a local user registry from some other user registry type."

"AMUUAU023W": "Uniqueness of names cannot be guaranteed when using the --sluggify-no-checksum option during unpack"  
For v10.0.5.1 and earlier: The archive:unpack command has been invoked with an option that could in unusual cases result in duplicate names being used for different unpacked entries. This is likely only for names with near-identical spelling but differ only in their use of upper and lower case letters or symbols that are converted to a hyphen during the unpack process. If duplicate names appear for entries of the same type in the same context, further messages indicating problems with the duplicate will appear.  
For v10.0.5.2 and later: If the migration utility is run on Windows or on Mac OS in a case-insensitive partition, additional errors may occur because of duplicate file names such as `name-1.yml` and `Name-1.yml`, which would be considered to be the same file."

"AMUUAU024E": "The stats summary could not be generated"  
The migration utility was unable to produce information listing the total numbers of entries of different types that were unpacked from the API Connect v5 migration archive. This message indicates an internal error in the migration utility."

"AMUUAU025I": "The {{.Type}} {{.Name}} contains an invalid value that will be removed."  
The migration utility has encountered an entry in the API Connect v5 migration archive that contains a value that is not valid for the property in which it appears. The invalid value is automatically removed. This can occur with values such as email or URL properties that contain values that do not match the required format."

"AMUUAU026I": "The {{.Type}} {{.Name}} contains an invalid value that will be replaced."  
The migration utility has encountered an entry in the API Connect v5 migration archive that contains a value that is not valid for the property in which it appears. The invalid value is automatically corrected. This can occur with values such as email or URL properties that contain values that do not match the required format."

"AMUUAU027W": "The product subscription {{.Name}} cannot be pushed to the target system."  
The migration utility has encountered a subscription entry in the API Connect v5 migration archive that refers to a product that is in a state that does not allow subscriptions. The subscription is removed and will not appear in the unpacked data."

"AMUUAU028E": "The migration tool failed to create default roles for {{.Type}} {{.Name}}."  
The migration utility has encountered an error while attempting to create the default roles for an organization, catalog, or space. This may be the result of a local file permissions problem or a shortage of space on the local filesystem."

"AMUUAU029W": "The OAuth provider {{.OAuthProvider}} with base path {{.BasePath}} could not be unpacked."  
The migration utility has failed to create an OAuth Provider with a particular base path. This can occur if there is an OAuth provider of this name with a different base path that has already been created in this provider organization. In API Connect v5, OAuth providers existed at the catalog level. In API Connect v10, they exist at the provider organization level, which can cause this issue to occur."

## Error messages and warnings from archive:push

"AMUAP001E": "The {{.Type}} {{.Name}} contains nested content that cannot be published."  
The migration utility is unable to read or load nested configuration content during the archive:push operation. After the migration unpack has been completed, some configuration objects are comprised of a directory with one or more nested files. For example, information about catalogs in their unpacked form consists of a directory with a name matching the catalog name, and a nested `catalog.yml` and `catalog-settings.yml` file. This message indicates that one or more of these files cannot be opened or loaded."

"AMUAP002E": "The {{.Type}} {{.Name}} cannot be published."  
The migration utility is unable to read or load configuration content during the archive:push operation. This message indicates that the file representing the named configuration object is unreadable or contains unexpected content."

"AMUAP003W": "Skipping mapping file {{.Name}}."  
The migration utility found a mapping file (`mapping.yml`) with invalid contents. The archive:push operation proceeds as if this mapping file were not present."

"AMUAP004E": "The file {{.Filepath}} could not be loaded and will not be published."  
The migration utility is unable to read or load configuration content during the archive:push operation. This message indicates that the file representing a configuration object cannot be read or loaded."

"AMUAP005E": "The file {{.Filepath}} contains references that could not be loaded and will not be published."  
The migration utility is unable to read or load configuration content during the archive:push operation. This message indicates that the file representing a configuration object cannot be read or loaded."

"AMUAP006E": "Unable to update reference to {{.Path}} in file {{.File}}."  
The migration utility is unable to resolve a reference in the named file to another configuration object. This message indicates that the content in the file refers to other objects that are not found in the unpacked content, or the referenced content is not of the expected format."

"AMUAP007E": "The {{.Type}} {{.Name}} cannot be published because of a missing prerequisite."

The migration utility is unable to publish the named object because it refers to another object that cannot be found in the target API Connect system. This message indicates that the object contains an incorrect reference, or that the referenced object has not been successfully created or has been deleted from the target system.

"AMUAP008W": "The update to the {{.Type}} {{.Name}} could not be verified and may not have been successful."

The archive:push operation has attempted to update an object in the target API Connect system, but the results of the update were not received or indicated an error. This can occur in the event of a network interruption or timeout between the migration utility and the target API Connect system. If this occurs, it is safe to retry the archive:push operation; the same update will be attempted.

"AMUAP009E": "The {{.Type}} {{.Name}} was not published because it could not be checked in the target system."

The migration utility has attempted to check the existence of an object in the target API Connect system, but the check did not complete successfully. The migration utility will not overwrite objects in the target system if it cannot successfully determine whether or not a conflict will occur.

"AMUAP010W": "The creation of the {{.Type}} {{.Name}} could not be verified and may not have been successful."

The migration utility has attempted to create an object in the target API Connect system, but the results of the creation were not received or indicated an error. This can occur in the event of a network interruption or timeout between the migration utility and the target API Connect system. If this occurs, it is safe to retry the archive:push operation; if the named object was created, then it will subsequently be skipped unless the overwrite option is used.

"AMUAP011W": "The deletion of the {{.Type}} {{.Name}} could not be verified and may not have been successful."

The migration utility has attempted to delete an object in the target API Connect system, but the results of the deletion were not received or indicated an error. This can occur in the event of a network interruption or timeout between the migration utility and the target API Connect system. If this occurs, it is safe to retry the archive:push operation.

"AMUAP012W": "The ownership transfer of the {{.Type}} {{.Name}} could not be verified and may not have been successful."

The migration utility has attempted to transfer ownership of an object in the target API Connect system, but the results of the ownership transfer were not received or indicated an error. This can occur in the event of a network interruption or timeout between the migration utility and the target API Connect system. If this occurs, it is safe to retry the archive:push operation.

"AMUAP013E": "The endpoint settings for catalog {{.Name}} could not be updated."

The migration utility has attempted to update the endpoint settings for a catalog in the target API Connect system, but the update was not successful. This can occur if the catalog settings refer to a gateway service for which the configuration cannot be found in the unpacked migration contents.

"AMUAP014E": "The {{.Type}} {{.NameVersion}} could not be {{.Action}} in the target system."

The migration utility has attempted to create or modify an object in the target API Connect system, but the target system returned an error. The reason associated with the message indicates the error returned by the server.

"AMUAP015E": "The gateway extension for the gateway service {{.Name}} could not be published."

The migration utility has attempted to publish a gateway extension zip to the target API Connect system, but the target system returned an error. The reason associated with the message indicates the error returned by the server.

"AMUAP016W": "The {{.Type}} for the gateway service {{.Name}} could not be updated."

The migration utility has attempted to update the OAuth shared secret or SNI items for a gateway service in the target API Connect system, but an OAuth shared secret or SNI items were previously set. The migration utility will not overwrite an OAuth shared secret or SNI items that have already been set.

"AMUAP017W": "The product {{.NameVersion}} could not be validated and may contain errors."

The migration utility has attempted to validate a product before publishing, but the validation operation did not complete successfully. If validation errors are present in the product file, then the publish to the target API Connect system will not be successful.

"AMUAP018W": "The product {{.NameVersion}} contains API {{.APIName}} that could not be validated and may contain errors."

The migration utility has attempted to validate the APIs in a product before publishing, but the validation operation did not complete successfully. If validation errors are present in the API file, then the publish to the target API Connect system will not be successful.

"AMUAP019W": "The product {{.NameVersion}} contains validation errors: {{.Errors}}"

The migration utility has validated the content of a product before publishing, and the validation revealed errors in the product file contents. The product cannot be published to the target API Connect system unless the validation errors in the product file are corrected.

"AMUAP020W": "The product {{.NameVersion}} contains an API {{.APINameVersion}} with validation errors: {{.Errors}}"

The migration utility has validated the content of a product and its associated APIs before publishing, and the validation revealed errors in the API file contents. The product cannot be published to the target API Connect system unless the validation errors in the API file are corrected.

"AMUAP021W": "The product {{.NameVersion}} contains API content that could not be updated."

The migration utility has attempted to update the state of an API in the named product, but the metadata file (\*.api.meta.yml) containing extra API attributes could not be read or contained unexpected content.

"AMUAP022W": "Subscription {{.Name}} will not be published."

The migration utility has encountered a subscription for which the subscribed product cannot be found. The subscription cannot be created in the target API Connect system.

"AMUAP023W": "The user registry {{.Name}} cannot be published."

The migration utility has encountered a user registry object that cannot be published to the target API Connect system. This can occur for user registry types that may appear in an API Connect v5 system but are not directly supported in API Connect v10. A mapping file in the unpacked user registry directory may be used to identify an alternative user registry into which user entries can be migrated.

"AMUAP024I": "Subscriptions for the product {{.NameVersion}} will not be published."

The migration utility has encountered a product that contains no subscribable APIs. This can occur when a product originating in API Connect v5 contains one or more OAuth provider APIs, and no REST or SOAP APIs. OAuth provider APIs are converted to OAuth provider resources by the archive:unpack operation, and do not appear as APIs in API Connect v10. OAuth provider instances in API Connect v10 use subscription information associated with the REST or SOAP APIs that refer to the OAuth provider in a security definition.

"AMUAP025E": "Non-fatal errors were found while running the preflight check: {{.Errors}}"

The migration utility encountered a non-fatal error during its preflight check before the start of **archive:push**. The details of any of these errors will be reported back here and the **archive:unpack** will continue.

## Troubleshooting tips

- The migration utility may have issues when running across a VPN. The connection might fail if you use a corporate VPN to attempt to access an API Connect subsystem (API Manager UI) located behind an internet firewall. To work around this problem, add the IP addresses of the API Connect management server and API manager to your local /etc/hosts.
- Token expiration: By default, the **apicm** login token expires after eight hours. If the token expires while **apicm archive:push** is running, **apicm** generates errors due to lack of the necessary access permissions. In this case, log in again and re-run **apicm archive:push**. (The token is refreshed every time you log in.) Optionally, you can extend the token expiration time when logging in as the admin user:

```
apicm login --expireHours=<number_of_hours>
```

The **expireHours** flag overrides the default authentication token expiration of 8 hours. Use this flag when pushing data sets that take longer than 8 hours to complete. Note that the revised value persists after the session ends. Best practice is to revert the value to the default (8 hours) after migration.

- If your prior migration created an incorrect artifact on v10, you may have to manually remove the artifact before re-running migration. If you use the `--overwrite` flag when running additional `archive:push` commands, existing OAuth Providers, draft products, draft APIs, and published products and their APIs will be overwritten.
- When you re-run migration, the utility will not recreate any artifacts that already exist on v10 unless the `--overwrite` flag is used for OAuth Providers, draft products, draft APIs, and published products and their APIs. If a collision occurs for other artifacts or if the `--overwrite` flag was not used, the older artifact takes priority. Also, the utility will not delete any artifacts from v10. Thus, you can re-run migration to cover any gaps in artifact creation on v10 without affecting existing artifacts.
- The steps you need to take depend on whether you need to change the v5 export data. For example, if you need to publish or replace a v5 API, you need to export the v5 data and run `unpack` again. If you do not need to change the v5 data, you can skip those steps.

#### Example 1: Re-using the v5 export data

For example, your v5 export data is accurate and current, but you migrated a consumer-org that is no longer needed on v10, and you mapped a catalog to an incorrect name. Complete the following steps:

1. Delete the incorrect artifacts from the v10 deployment `/cloud` directory. For this example, delete the unneeded consumer-org and the incorrectly named catalog.
2. Edit the contents of the `/cloud` directory as needed. For example, remove the unneeded consumer-org entry.
3. Edit your mapping entries as needed. For example, remap the catalog name.
4. Run `apicm archive:push <server> <project_directory>`.

#### Example 2: Revising v5 export data

In this example your v5 export data is not current because you want to publish or replace a v5 API and then migrate it to v10. Complete the following steps:

1. On v5, recreate the export data: `config dbextract sftp <host_name> user <user_name> file <path/name>`
2. On v10, delete the `/cloud` directory.
3. Run `apicm archive:unpack` to create the v10 hierarchy in `/cloud` directory.
4. Recreate any edits you made to the `/cloud` directory.
5. Recreate any mappings.
6. Run `apicm archive:push <server> <project_directory>`.

## Failure to convert API definitions to API Gateway

When you change gateway types from DataPower Gateway (v5-compatible) to DataPower API Gateway, you use `./apicm archive:port-to-apigw <path>` to convert API definitions to API Gateway format. The conversion requires Node version 18.17.1 or greater. If Node is not installed, this error is displayed:

```
ERRO[0000] Error getting node version: exec: "node": executable file not found in $PATH
```

If an unsupported version of Node is installed, an error is displayed as in the following example:

```
ERRO[0000] Node version installed must be >=18.17.1. Currently installed version: 4.9.1. Use the --allow-older-node flag to allow the use of an older node 18 version at the users own risk.
```

To resolve the error, install Node and re-run the command. If you are on an older version of node 18 you can re-run the command using the `--allow-older-node` flag but you do this at your own risk.

To view the conversion step, see [Migration steps without PDUR](#) or [Migration steps with PDUR](#). For information about the conversion, see [Reviewing APIs converted to DataPower API Gateway](#).

## Custom policies not appearing in the policy palette after migration

If you are migrating to v10.0.1.3 or earlier, or v10.0.2.0 or earlier, follow these steps to have custom policies appear in the policy palette in the API Manager Assembly UI.

1. Open `gateway-extension.zip` in the `cloud/admin-org/.../<gateway-service>` folder that is generated in `admin-org` after `port-to-apigw` is run.
2. Open the `manifest.json` file.
3. Edit the `manifest.json` file to specify the implementation files for the custom policies to push to the sandbox catalog. Add entries for the sandbox catalog for all provider-orgs using the format `<provider-org>_<catalog-name>`. The filename can remain the same for each respective provider-org, as it will be the same custom policy. Ensure that the actual file that the filename is pointing to in the `.zip` file exists. The same filename can be re-used multiple times in the `manifest.json` to add the sandbox entries for the custom policy. Only the `manifest.json` must be edited. You do not need to edit any of the `.zip` files in the `gateway-extension.zip` package.

4. Save the `manifest.json` file and zip it.
5. If the `archive:push` was already performed before step 1, then delete the `gateway-extension.zip` file from the gateway using the API Connect toolkit command `apic gateway-extensions:delete`.
6. Push `admin-org`, which pushes the updated `gateway-extension.zip` file.
7. Follow these steps to ensure that the API Manager advertises the custom policy that is being pushed.
  - a. In each of the provider orgs in the cloud folder, navigate to the `catalog-scoped-staging` folder and log in to the API Connect toolkit as a provider-org user.
  - b. Run this command on every `.zip` file that you want to appear in the policy palette in the v10 UI:

```
apic policies:create -c sandbox --configured-gateway-service -o -s $API_SERVER_URL --scope catalog .zip
```

The `gateway-service` name is under Topology. `$API_SERVER_URL` is the same one you use to log in to the API Connect toolkit. The custom policy zip filename will appear in the `catalog-scoped-staging` folder.

8. Navigate to the API Manager UI and check the assembly policy palette. The custom policies should now appear in the list. You do not need to push the provider orgs.

If you are migrating to v10.0.1.4 or later, v10.0.3.0 or later, follow these steps to have custom policies appear in the policy palette in the API Manager Assembly UI.

1. After the migration of `admin-org` and `provider-orgs` has completed, navigate to the API Manager UI.
2. While editing a draft API, click the gear icon next to the offline button. A pop-up should appear.
3. Select the `target-catalog` to publish the draft API to. The `target-catalog` must already have the custom policies pushed to it.
4. Select the Application to use for the API.

5. Click Save.
6. Navigate to the Assembly policy palette. The custom policies for that specific target-catalog should now appear. For more details about these steps, see [Specifying the testing preferences for an API](#).

When artifacts such as provider org, space or catalog are mapped, any warning or error messages for artifacts in subdirectories underneath those may have the location incorrect. The location will include the mapped artifact location instead of the original location.

---

## Migrating from v10 to v10 on a different form factor

Migrate an API Connect v10 deployment from one form factor to a different form factor.

- [v10 migration overview](#)  
Use migration scripts to migrate API Connect to another form factor, or to a different deployment on the same form factor.
- [v10 migration requirements](#)  
Ensure your deployment meets the migrations prerequisites and requirements.
- [v10 migration steps](#)  
Complete the steps that are required for your v10 migration path.
- [v10 migration troubleshooting](#)  
Troubleshoot problems with v10 migration.
- [Shutting down the Postgres database](#)  
Shut down the Postgres database to ensure no changes are made to the database during migration.

---

## v10 migration overview

Use migration scripts to migrate API Connect to another form factor, or to a different deployment on the same form factor.

Scripts are run from a client machine which has access to the Kubernetes or OpenShift cluster. Scripts are supported on Linux and macOS.

It is recommended to review all the documentation before starting a v10 migration.

Overview topics:

- [Migration terminology used in this documentation](#)
- [Location of v10 migration scripts](#)
- [Supported v10 migration paths](#)
- [v10 migration planning](#)

---

## Migration terminology used in this documentation

This documentation uses the following terms for migrations:

- v10 migration  
The term **v10 migration** includes:
  1. Migration of a v10 API Connect deployment to a v10 API Connect deployment on a different platform.
  2. Migration of a v10 API Connect deployment to a v10 API Connect deployment in a different cluster on the same platform.  
Applicable to native Kubernetes, OpenShift, and Cloud Pak for Integration.
  3. Migration of a v10 API Connect deployment to a Cloud Pak for Integration deployment.
  4. Migration of a Cloud Pak for Integration deployment to a Cloud Pak for Integration deployment on a different cluster.  
Note that Cloud Pak for Integration is deployed only on the OpenShift platform.
- Form Factor and Form Factor Migration.  
The term **Form Factor** is used in v10 migration to characterize a cluster deployment on a particular platform.  
  
Each of the migrations included in the definition for **v10 migration** is a movement of an existing v10 deployment to a different **Form Factor**.  
  
The term **Form Factor Migration** is another term for **v10 migration**. It refers to different software deployments. Usage of the term **form factor** in this documentation does not refer to any hardware form factors.
- Platform  
The term **platform** is used to mean either native Kubernetes, VMware virtual machine, OpenShift Container Platform, or Cloud Pak for Integration.
- Migration  
In the v10 API Connect documentation, the term **migration** is also used for scenarios for migrating from API Connect v5 to API Connect v10. Note that migrating from v5 to v10 is unrelated to moving a v10 deployment to a different form factor.  
  
To view instructions for migrating from v5 to v10, see [Migrating a Version 5 deployment](#).

---

## Location of v10 migration scripts

You can obtain the scripts from Fix Central. Obtain the following package:

```
apiconnect-operator-release-files_VERSION_NUMBER/helper_files/formFactorMigration
```

To access the latest version, see the [What's new in the latest release](#) page. On that page, locate the Note: *You can access the latest files from <URL link>*. Select the <URL link> to go directly to the Announce page on Fix Central, where you can download files for the latest version of API Connect.



## Supported v10 migration paths

The supported migration paths cover two use cases:

- Migrating from a source system on one form factor to a target system on another form factor.
- Migrating from a source system on one form factor to a target system on the same form factor, when there is a change in hostnames.

Important: The API Connect version on the target system must match the API Connect version on the source system.

Table 1. Supported migration paths for API Connect v10

| Source deployment platform of API Connect v10 | Target deployment platform, based on API Connect v10                                       |
|-----------------------------------------------|--------------------------------------------------------------------------------------------|
| Kubernetes                                    | Cloud Pak for Integration                                                                  |
| Kubernetes                                    | OpenShift Container Platform                                                               |
| Kubernetes                                    | Kubernetes, when moving to a different v10 cluster, resulting in hostname changes          |
| VMware                                        | Cloud Pak for Integration                                                                  |
| VMware                                        | OpenShift Container Platform                                                               |
| OpenShift Container Platform                  | Cloud Pak for Integration, using the same hostnames                                        |
| OpenShift Container Platform                  | Cloud Pak for Integration, moving to a different cluster, resulting in hostname changes    |
| OpenShift Container Platform                  | OpenShift Container Platform, moving to a different cluster, resulting in hostname changes |
| Cloud Pak for Integration                     | Cloud Pak for Integration, moving to a different cluster, resulting in hostname changes    |

## v10 migration planning

Consider the following information when planning a v10 migration:

- **Data migration**

There are two ways to migrate data:

- Perform complete migration of data:

Before starting migration steps on the target system, have the target system ready with the same or larger topology than what is present in the source system. This means the number of Gateways, Portals, and Analytics subsystems must be same or more than what is present in the source system. Start the migration process, map the endpoints between source and target systems, and complete the migration process. All data is migrated. The target system must be validated and tested.

If you do not want to migrate data for some subsystems on the source system, skip those endpoints in the mapping. Then, after migration is complete, manually delete the data from Cloud Manager and the API Manager UI on the source system.

- Perform migration in more than one step

In this scenario, at the time of migration the API Connect target system does not have all the subsystems ready when compared to the total number of subsystems in the source system. Migration is run only for the Gateway and Portal subsystems that are available on the target system. At this point, the target system will still have some stale data related to the subsystems that are not migrated. You can then create additional subsystems (Gateways, Portals, or Analytics) on the target system, and run the necessary migration steps to complete the remaining migration process.

- If additional Gateways or Analytics are added to the source system after completing the initial migration, you must:
  1. Add the new Gateway and Analytics related endpoints in the mapping file.
  2. Register Gateways and Portals on the target system. Complete Step 3 in [Restoring the target system](#).
  3. Update the gateways in the catalog to point to the new Gateway systems. Complete Step 6 in [Restoring the target system](#).
- If additional Portals are added to the source system after completing the initial migration, you must:
  1. Add the new Portal related endpoints in the mapping file.
  2. Register gateways and portals on the target system. Complete Step 3 in [Restoring the target system](#).
  3. Update the portals in the catalog to point to the new portal systems. Complete Step 4 in [Restoring the target system](#).
  4. Restore portal backups for each site on the namespace where the new portal is installed. Complete Step 5 in [Restoring the target system](#).
- The stale data for which migration is not complete will remain on the target system until you install the additional subsystems and complete migration, or until you use the Cloud Manager or API Manager UI to manually delete it.

- Reuse of Gateways from the source system:

1. Export the `ingress-ca` secret from the source system using the `-export_cert` flag in [Preparing the source system](#). The secret is saved to `data/MANAGEMENT_SUBSYSTEM_NAME/ingress-ca_secret.yaml`.
2. Install the `ingress-ca` secret manually on the target cluster. This step must be completed before you begin [Installing the target system](#).
  - If using top-level CR on OpenShift: The top-level CR name you are going to use while installing API Connect must be used while creating the secret name and its contents.
  - The name of the `ingress-ca` secret must be `TOP_CR_NAME-ingress-ca` in case of OpenShift using top-level CR, or CP4I. For Kubernetes or OpenShift using individual subsystems, it must be `ingress-ca`.
  - An example of an `ingress-ca` secret is given in the following example. If top-level CR is used, replace `TOP_CR_NAME` with the top-level CR name you are going to use to install API Connect. For Kubernetes or OpenShift using individual subsystems, all the values will be just `ingress-ca`.
  - After modifying the contents of `ingress-ca` and saving the yml, create the secret using `oc create -f filename.yaml -n NAMESPACE`.

```
apiVersion: v1
data:
 ca.crt: CA_CERT
 tls.crt: TLS_CERT
 tls.key: TLS_KEY
kind: Secret
metadata:
 annotations:
 cert-manager.io/alt-names: ''
```

```
cert-manager.io/certificate-name: TOP_CR_NAME-ingress-ca
cert-manager.io/common-name: ingress-ca
cert-manager.io/ip-sans: ''
cert-manager.io/issuer-group: ''
cert-manager.io/issuer-kind: Issuer
cert-manager.io/issuer-name: TOP_CR_NAME-self-signed
cert-manager.io/uri-sans: ''
name: <TOPCR>-ingress-ca
type: kubernetes.io/tls
```

3. Run the migration steps, starting with Step 1 in [Installing the target system](#). Install API Connect, using the same TOP\_CR\_NAME (if using top-level CR), by following the remainder of the steps in [Installing the target system](#).
  4. When you complete [Restoring the target system](#), do not map the source system gateways that are to be reused. Just skip them.
  5. Once the migration is complete, restart the gateways pods on the source system, wait for few minutes, and run the health check script to check the status of webhooks.
  6. Validate the APIs running on the source system gateways.
- Delta changes

Once the migration procedure is started, any delta changes in the source system databases are not handled by the migration scripts. These changes must be handled manually.

    - To avoid delta changes, shut down the Postgres database in the management subsystem. See [Shutting down the Postgres database](#).
    - If there are delta changes to the management database after saving the configuration for the source system, you can optionally take another backup of the Management subsystem, just prior to restoring the Management subsystem. See [Restoring the target system](#).
  - Sizing

Before installing the API Connect target system, make sure proper capacity planning is done to handle the production workloads. Ensure you have sufficient memory, CPU, and disk size as needed, based on the deployment profile being installed.
  - Expected migration time

The time required to complete migration can depend on the following:

    - The size of the management database. Backup and restore are dependent on this size.
    - Migration of the Portal and Gateway data in all the provider orgs depends on:
      - Number of provider orgs
      - Number of catalogs in these provider orgs
      - Whether Portals are configured for these catalogs
      - Total number of APIs published
    - Creation of Portal sites and restore of Portal backup depends on:
      - Deployment profile (n1, n3 etc) being used
      - Disk performance
      - Total number of Portal sites being create or restored
      - The amount of customization in a Portal site. Customization plays a role in the time needed to restore.

---

## v10 migration requirements

Ensure your deployment meets the migrations prerequisites and requirements.

Topics:

- [Version prerequisites](#)
- [Subsystem requirements](#)
- [Scripting requirements](#)
- [Credentials requirements](#)
- [How to obtain hostnames](#)
- [How to obtain realm values](#)
- [Backing up subsystems](#)
- [Limitations](#)

---

## Version prerequisites

The API Connect versions on the source and target systems must be the same.

---

## Subsystem requirements

All subsystems must be in a healthy state.

You must manually create storage classes like block storage or file system storage. Applies to all subsystems and, when installing on CP4I, to Platform Navigator.

Ensure that any network route between the target installation and the source system is disabled. The target system must not be able to communicate with the source system.

- Management subsystem:
  - You must configure Management subsystems with either S3 or SFTP backups.
  - If s3 storage is being used, postgres in the Management subsystem must be stopped on the source system.

Note:

Shutting down postgres is necessary in this scenario because two APIC deployments cannot both use the same S3 backup location. The target API Connect deployment will not start if postgres on the source deployment is up and connecting to the same S3 backup location.

When postgres is stopped, there is no impact on the published APIs in the gateways. However, the Cloud Manager and API Manager UI will not be available because the underlying microservices are down. Users are able to browse the APIs in the Portal. But new consumer orgs or applications or users cannot be created.

- If the Postgres cluster name on the target is different than the source system, restore will fail. For more information on configuring the target system, see [Installing the target system](#).
- Developer Portal subsystem:
  - Developer Portal subsystems must be configured with either S3 or SFTP backups.
  - For Developer Portal subsystem, you must have all the mappings between old and new endpoints and hostnames.
- Gateway subsystem:
  - If a custom gateway image is being used on the source system, the same image must be used on the target system
  - For a Gateway subsystem, you must have all the mappings between old and new endpoints and hostnames.

## Scripting requirements

- Python 3 and PyYAML python module must be present on the system where the scripts are run.  
To install the PyYAML module:
 

```
pip3 install pyyaml
```
- Do not have any spaces in the complete directory path where the scripts are executed.
- Ensure that the API Connect version of the **apic** toolkit is the same version as for the scripts, and that the toolkit is located in the path where the scripts are executed. Validate toolkit commands to make sure the Cloud Manager and API Manager hostnames are resolved and able to run few commands. Make sure apic toolkit commands are working before trying out the scripts.
- If you are using hostnames, you may need to update the `/etc/hosts` file for the apic toolkit to work.
- When the scripts are run on the target system, the configuration (data directory) saved from the source system must be available in the location where scripts are present.
- Save the output logs for every script being executed.

## Credentials requirements

- In all the scripts where the credentials are passed on the command line, these credentials are the admin org (Cloud Manager) credentials.
- You must be able to use **kubectl** (for native Kubernetes) or **oc** (OpenShift) commands to access the source system cluster and the target system cluster.
- To administer the admin org, you must have credentials with administrative role to access Cloud Manager.
- For provider orgs, if you do not use the migration user, and plan to use provider org credentials, you must have the provider org credentials for the API Manager interface.
- You must manually create:
  - Secrets for entitlement keys to download images from Docker registry.
  - DataPower admin credentials and secret, if the target system is native Kubernetes.
- After migration, the users on the target system will be in the same user registry as in the source system. For example, in CP4I, users from a local user registry (LUR) are not moved to a Zen user registry.

## How to obtain hostnames

The hostnames used in the scripts (`--server` or `-s` or `-api_manager_hostname`) and also in the files `gateway_portal_mapping.yaml` or `provider_org_credentials.yaml` can be obtained using the following commands.

The command must be run on the source cluster or target cluster, based on the step for which the hostname is needed.

- Kubernetes:
 

```
kubectl get ingress -n <NAMESPACE>
```
- OpenShift and Cloud Pak for Integration:
 

```
oc get routes -n <NAMESPACE>
```

Example output on Kubernetes:

```
kubectl get ingress -n ask1
NAME CLASS HOSTS ADDRESS
PORTS AGE
analytics1-ac-endpoint <none> ac1.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166 80, 443 5d7h
analytics1-ai-endpoint <none> ail.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166 80, 443 5d7h
apigw1-gateway <none> rgw1.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166 80, 443 5d5h
apigw1-gateway-manager <none> rgwd1.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166 80, 443 5d5h
management-admin <none> admin.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166 80, 443 5d7h
management-api-manager <none> manager.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166 80, 443 5d7h
management-consumer-api <none> consumer.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166 80, 443 5d7h
management-platform-api <none> api.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166 80, 443 5d7h
portall-portal-director <none> api.portall.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166 80, 443 5d7h
portall-portal-web <none> portall.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166 80, 443 5d7h
v5gw1-gateway <none> gw1.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166 80, 443 5d5h
v5gw1-gateway-manager <none> gwd1.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,
```

## How to obtain realm values

The value of realm (`--realm` or `-r`) used while running the scripts can be obtained as shown in the following examples. The realm values are for Cloud Manager (admin org) and provider org.

- Example for Cloud Manager (admin org):  
The value is passed as a command line option while running the script.

```
apic identity-providers:list --scope admin --fields title,realm --server api.test1-master.fyre.example.com
total_results: 3
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: googleoidc
 realm: admin/googleoidc
- title: ldap1
 realm: admin/ldap1
```

- Example for provider org realm values:  
The value is used in `provider_org_credentials.yaml` while updating Portal and Gateway information in the provider orgs, when the default migration user is not used. See Step 4 and Step 6 in [Restoring the target system](#).

```
apic identity-providers:list --scope provider --fields title,realm --server api.testuser1-master.fyre.example.com
total_results: 3
results:
- title: API Manager User Registry
 realm: provider/default-idp-2
- title: googleoidc
 realm: provider/googleoidc
- title: ldap1
 realm: provider/ldap1
```

## Backing up subsystems

---

It is necessary to backup the data in Management and Portal subsystems on the source system. If backup is not configured, you can use the following links to configure backup:

- Management subsystem. Use the directions for your platform:
  - Native Kubernetes: [Backing up and restoring the management subsystem](#)
  - OpenShift and Cloud Pak for Integration: [Backing up and restoring the management subsystem on OpenShift and Cloud Pak for Integration](#)
  - VMware: [Backing up and restoring the Management subsystem](#)
- Developer Portal subsystem. Use the directions for your platform:
  - Native Kubernetes: [Backing up and restoring the Developer Portal in a Kubernetes environment](#)
  - OpenShift and Cloud Pak for Integration: [Backing up and restoring Developer Portal on OpenShift and Cloud Pak for Integration](#)
  - VMware: [Backing up and restoring the Developer Portal](#)

## Limitations

---

1. Analytics service and data is not migrated.
2. If migrating from v10 non-CP4I environment (source apic cluster) to v10 CP4i environment (target apic cluster), make sure the Cloud Manager password for `admin` user is not `7iron-hide` on the source cluster. If default password `7iron-hide` is used, migration will fail after step 3 in [Restoring the target system](#), as configurator job assumes that the install is a fresh installation, and changes the password automatically.

## v10 migration steps

---

Complete the steps that are required for your v10 migration path.

## Before you begin

---

Ensure you have reviewed [v10 migration overview](#) and [v10 migration requirements](#).

## Procedure

---

1. Identify the supported migration path you will use. If necessary, review [Supported v10 migration paths](#) in [v10 migration overview](#).
2. Follow the instructions for your migration path:
  - For all paths except migration from OpenShift top-level CR to Cloud Pak for Integration on the same cluster, complete the following tasks in order:
    - [Preparing the source system](#)
    - [Installing the target system](#)
    - [Restoring the target system](#)
    - [Verifying the target system](#)
  - If migrating from OpenShift top-level CR to Cloud Pak for Integration on the same cluster, complete [Migrating from OpenShift to Cloud Pak for Integration on the same cluster](#).
- [Preparing the source system](#)  
Run the `save_v10_source_configuration.py` script to gather the source system configuration.
- [Installing the target system](#)  
On the target system, configure a cluster, create the necessary secrets, and install API Connect.
- [Restoring the target system](#)  
Restore the saved configuration and backups from the source system onto the target system.
- [Verifying the target system](#)  
Verify that the target system is in a healthy state and fully configured.

- [Migrating from OpenShift to Cloud Pak for Integration on the same cluster](#)  
Migrate your API Connect v10 deployment from OpenShift to Cloud Pak for Integration, while remaining on the same cluster.

## Preparing the source system

Run the `save_v10_source_configuration.py` script to gather the source system configuration.

### Before you begin

Review [v10 migration requirements](#).

### About this task

The `save_v10_source_configuration.py` script:

- Creates a data directory with all configuration saved in it.
  - Generates two YAML files in the same directory as the script:
    - `gateway_portal_mapping.yaml`
    - `provider_org_credentials.yaml`
- Important:** The YAML files must be filled in:
- `gateway_portal_mapping.yaml` must be filled in if, when restoring onto the target system, you choose to run `register_gateway_portals_in_target.py` in silent mode. See step 3.c in [Restoring the target system](#).
  - `provider_org_credentials.yaml` must be filled in if, when restoring onto the target system, you choose not to use the migration user credentials while updating Portal and Gateway information. See step 4 and step 6 in [Restoring the target system](#).

To view script parameters:

```
python3 save_v10_source_configuration.py --help
```

## Procedure

1. Save configuration from the source system. Complete the instructions for the platform type of your source system:

- Kubernetes, or OpenShift, or Cloud Pak for Integration

To save the source system configuration, run `save_v10_source_configuration.py` with the appropriate parameters for your deployment:

- If all subsystems are in the same namespace, and you are using username/password authentication. This command saves configuration of all the installed subsystems in the namespace:

```
python3 save_v10_source_configuration.py -u USERNAME -p PASSWORD -s PLATFORM_API_HOSTNAME -r REALM_NAME -n NAMESPACE
```

- If a subsystem needs to be skipped so that its configuration is not saved:

- Skip Portal:

```
python3 save_v10_source_configuration.py -u USERNAME -p PASSWORD -s PLATFORM_API_HOSTNAME -r REALM_NAME -n NAMESPACE -skip_ptl
```

- Skip Analytics:

```
python3 save_v10_source_configuration.py -u USERNAME -p PASSWORD -s PLATFORM_API_HOSTNAME -r REALM_NAME -n NAMESPACE -skip_a7s
```

- If all subsystems are in the same namespace, and you are using SSO authentication:

```
python3 save_v10_source_configuration.py --sso -s PLATFORM_API_HOSTNAME -n NAMESPACE
```

- If all subsystems are in different namespace, and you are using username/password authentication:

```
python3 save_v10_source_configuration.py -u USERNAME -p PASSWORD -s PLATFORM_API_HOSTNAME -r REALM_NAME -mgmt_ns <mgmt_namespace> -ptl_ns <ptl_namespace> -gw_ns <gw_namespace> -a7s_ns <a7s_namespace>
```

- If the cluster has multiple Gateways, Portals, or Analytics subsystems distributed across namespaces in the cluster, you can save the configuration of the source system using one of the ways shown below. Multiple namespaces need to be separated by "|" symbol. The namespaces used in below example command are `ns1`, `ns2`, `ns3`. Both commands work the same way.

- `python3 save_v10_source_configuration.py -u USERNAME -p PASSWORD -s PLATFORM_API_HOSTNAME -r REALM_NAME -n "ns1|ns2|ns3"`

- `python3 save_v10_source_configuration.py -u USERNAME -p PASSWORD -s PLATFORM_API_HOSTNAME -r REALM_NAME -mgmt_ns "ns1" -ptl_ns "ns1|ns2" -gw_ns "ns1|ns2|ns3" -a7s_ns "ns1|ns2"`

- If ingress-ca certificates must be saved or exported in the configuration, run the previous commands with `-export_cert` flag. Applicable only when you are saving management subsystem configuration.

- VMware

On VMware appliance, save the source configuration by running the `save_v10_source_configuration.py` script on the Management and Portal nodes.

If you have a three replica deployment, run the `save_v10_source_configuration.py` script on just one of the VMs in the cluster.

- Copy the `apic` toolkit and `apicup` installer to the management node.
- Copy the script `save_v10_source_configuration.py` to the management node:

- If `ingress-ca` and `root-ca` certificates must be saved in the configuration, copy this script to the project directory for this appliance and upload the project directory to the appliance. The script must be run from inside the project directory.
- If `ingress-ca` and `root-ca` certificates are not needed, just copy this script to the appliance.
- Change to root user (`sudo -i`) and set PATH variable to the directory where the `apic` toolkit is present. This directory must be at the beginning of the path.
- Change to the directory where script is located. Check the PATH to verify that the `apic` toolkit is present. To validate, use `apic login` and `apicup version`.
- Collect source system configuration. If certificates needed, add the flag `-export_cert` to the command.

```
python3 save_v10_source_configuration.py -u USERNAME -p PASSWORD -s PLATFORM_API_HOSTNAME -r REALM_NAME -mgmt_ns default
```

- On the Management node, from the location where the script is run, download the data directory, `gateway_portal_mapping.yaml`, and `provider_org_credentials.yaml` files. Copy them to the location where the form factor migration scripts are present.
- Copy the `save_v10_source_configuration.py` script to the Portal node and run the script as root user, using the following command:

```
python3 save_v10_source_configuration.py -ptl_ns default -skip_mgmt
```

On the Portal node, from the location where the script is run, download the `data` directory and merge it with the `data` directory downloaded from the Management node.

If multiple Portal services are configured with the Management subsystem, repeat the same operation for each Portal.

2. Next, begin configuration of the target cluster. Continue with [Installing the target system](#).

---

## Installing the target system

On the target system, configure a cluster, create the necessary secrets, and install API Connect.

### Before you begin

- Complete [Preparing the source system](#).
- Ensure that any network route between the target installation and the source system is disabled. The target system must not be able to communicate with the source system.

### About this task

Run the `create_secrets_in_target.py` script to create the necessary secrets. If necessary, shutdown the Postgres database, and then install API Connect, either manually or by running the `install_apic_on_ocp.py` script.

Note: For data migration, you have the option of performing a complete migration at once, or performing migration in more than one step. To review these options, see [v10 migration planning](#).

### Procedure

1. Configure a target cluster either Kubernetes or OpenShift or Cloud Pak for Integration, where you intend to migrate the source system data.
  - Note that this configuration is for the cluster only. No API Connect software is installed yet.
  - Make sure `KUBECONFIG` is set to this target cluster, so you can access the cluster using `oc` or `kubectl`.
2. Create the namespaces in the target cluster where you want the API Connect subsystems to run.
  - The namespaces can be different than the source system values.
  - If you have additional subsystems to be installed in different namespaces, create these namespaces also.
3. Run `create_secrets_in_target.py` to create the following secrets:
  - Database backup secrets for Management and Portal subsystems
  - Application credentials
  - Encryption secrets from the source system

Usage examples:

  - If both Management and Portal are in the same namespace, or if Portal is not installed.
 

```
python3 create_secrets_in_target.py -n NAMESPACE
```
  - If both Management and Portal are in different namespaces:
 

```
python3 create_secrets_in_target.py -mgmt_ns <MGMT_NAMESPACE> -ptl_ns <PTL_NAMESPACE>
```
  - If you are creating multiple Portal subsystems in the target API Connect system, to match the configuration of the source system, you must run the script more than once for each Portal subsystem from the source system. The script detects that there are two Portal subsystems in the source system, and prompts you to select the subsystem name that matches the target Portal, as shown in the following commands:
    - For Management and first Portal on the target system:
 

```
python3 create_secrets_in_target.py -mgmt_ns <MGMT_NAMESPACE> -ptl_ns <PTL_NAMESPACE1>
```
    - For the second Portal on the target system:
 

```
python3 create_secrets_in_target.py -skip_mgmt -ptl_ns <PTL_NAMESPACE2>
```
4. If applicable to your deployment, shutdown the Postgres database on your source system. You can shutdown the Postgres database now, to ensure that there are no delta changes to your source system while the remainder of the migration process takes place.

Important: If s3 backup is being used for the Management subsystem in the source system, the postgres database MUST be shutdown before launching the target API Connect cluster. If it is not shutdown, the launch of the target apic cluster will fail.  
For steps to shutdown Postgres database on Kubernetes or OpenShift, see [Shutting down the Postgres database](#).

## 5. Launch the target API Connect system.

### a. Review prerequisites:

- Have a new OpenShift cluster ready with enough resources (CPU, memory, and disk) as required for the installation profile you will use.
- As needed, create an entitlement key to download the images related to API Connect.
- The target system must match the topology (number of Gateways, Portals and Analytics) of the source system to ensure that all the data in the Management database is migrated. If all the data in the Management database is not migrated, the not migrated (stale) data must be manually deleted. Alternatively, you can attempt the remaining migration later after installing the additional Portals and Gateways. See [v10 migration planning](#).

Important: If s3 backup is being used for the Management subsystem in the **target** system, once the Management subsystem is ready, you will see that the backup job pods and stanza-create job pods are in **Error** state. This is expected behavior. Once management backup is restored, the error resolves and backups will work. Since the backup will **not** happen until the Management database is restored, it is very important to restore Management subsystem backup on the target system (see step 2 in [Restoring the target system](#)) immediately after installing the target system. Any delay can cause the disk to fill up with write-ahead log (WAL) files.

To review how to monitor Postgres db disk usage, see:

- [Monitoring Postgres disk usage on OpenShift](#)
- [Monitoring Postgres disk usage on VMware](#)
- [Monitoring Postgres disk usage on Kubernetes](#).

### b. Launch the target API Connect system either manually or by using the `install_apic_on_ocp.py` script:

- To manually launch from the OpenShift cluster UI:

This method is the recommended way to launch the target API Connect system for production systems.

This method is also used for advanced customization or when using custom certificates while launching the target API Connect system.

- Install and launch the target API Connect cluster. See the instructions for your deployment type:
  - API Connect on OpenShift: [Deploying on OpenShift and Cloud Pak for Integration](#)
  - Cloud Pak for Integration: [Installation procedures with IBM Cloud Pak for Integration](#)
- Before creating the top level CR, use the yaml view to add the properties which you saved from the source system configuration. For the Management subsystem, the saved configuration is in `data/config.yaml`. For the Portal subsystem, the saved configuration is in `data/config_portal.yaml`.

Important: If you do not use the configuration from the source system, the migration fails.

- Management subsystem properties:
  - encryptionSecret
  - Management subsystem name
  - site name
  - database backup credentials
  - custom application credentials
  - originalUID
- Portal subsystem properties:
  - encryptionSecret
  - site name
  - database backup credentials
  - originalUID

Note: If target system is Kubernetes, the properties must be in the Management and Portal subsystem CRs.

- The top level CR must look like the following example. The values for this top CR are from the generated artifacts (`data/config.yaml` and `data/config_portal.yaml`) after running [Preparing the source system](#). Note that the example shows a value for `spec.management.name`. You can obtain this value from `config.yaml`.

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APICoconnectCluster
metadata:
 labels:
 app.kubernetes.io/instance: apiconnect
 app.kubernetes.io/managed-by: ibm-apiconnect
 app.kubernetes.io/name: apiconnect-minimum
 name: c1
 namespace: test1
spec:
 analytics:
 storage:
 enabled: true
 type: unique
 license:
 accept: true
 license: SOME_KEY
 use: nonproduction
 management:
 customApplicationCredentials:
 - name: atm-cred
 secretName: management-atm-cred
 - name: ccli-cred
 secretName: management-ccli-cred
 - name: cui-cred
 secretName: management-cui-cred
 - name: dsgr-cred
 secretName: management-dsgr-cred
 - name: juhu-cred
 secretName: management-juhu-cred
 - name: cli-cred
 secretName: management-cli-cred
 - name: ui-cred
```

```

 secretName: management-ui-cred
 databaseBackup:
 credentials: ftp-mgmt-backup-secret
 host: SFTP_SERVER_HOST
 path: /root/backup/v10_management
 port: 22
 protocol: sftp
 restartDB:
 accept: false
 encryptionSecret:
 secretName: management-enc-key
 name: management
 originalUID: d435b534-ce93-463a-9005-692c2af94df7
 siteName: be023200
portal:
 encryptionSecret:
 secretName: portall1-enc-key
 originalUID: 1adc672a-0e77-4890-b476-c41fcedb0b64
 portalBackup:
 credentials: ftp-mgmt-backup-secret
 host: 1.2.3.4
 path: /root/backup/portall1
 port: 22
 protocol: sftp
 siteName: cb06099f
 profile: n1xc7.m48
 storageClassName: rook-ceph-block
 version: 10.0.4.0

```

- You can also use the script `install_apic_on_ocp.py` with `-no_install` flag to generate the top level CR yaml, and use the yaml in the OpenShift UI and customize further.
- To launch using the `install_apic_on_ocp.py` script:  
The script is used to launch API Connect on OpenShift or API Connect with Cloud Pak for Integration using a top level CR that specifies one of each subsystem type:
  - Management
  - Developer Portal
  - Analytics
  - API Gateway

Prerequisite:

- The OpenShift cluster must be ready, and accessible using `oc`.
- Important:** The script is used to launch **ONLY** the specific version of API Connect based on the downloaded version of the artifacts.

Usage:

- Run the script with `--help` option to show additional flags available like `-production` and `-profile`.
- License use is set to `nonproduction` by default unless the flag `-production` is used while running this script.
- The profile defaults to the one replica profile `n1xc7.m48`, to specify a different profile use the `-profile` flag. For details of the available profiles, see [API Connect deployment profiles for OpenShift and Cloud Pak for Integration](#). The one replica profile deploys 1 replica of each pod and is only for light, non-critical workloads such as basic development and testing.
- Optionally, you can use the `-no_install` flag if you want to generate the yaml files (top level CR and other files needed in installation) without installing. In the following example commands, add the `-no_install` flag to the existing command parameters. You can use this top level CR from the OpenShift UI, and create an API Connect cluster.
- Note : If CP4I environment, check if configurator job is completed and also check the output of:

```
oc get apiconnectcluster -n NAMESPACE
```

Example command invocations:

Note: Each of the example commands includes parameters for a three replica profile deployment: `-production -profile n3xc16.m48`. If you have a one replica profile, omit these parameters.

- API Connect on OpenShift:
  - If the apiconnect and DataPower operators must be installed in all namespaces:

```
python3 install_apic_on_ocp.py -license LICENSE_VALUE -n NAMESPACE -name topCRName -storageclass_apic STORAGE_CLASS_NAME -production -profile n3xc16.m48
```

- If the apiconnect and DataPower operators must be installed in a specific namespace:

```
python3 install_apic_on_ocp.py -license LICENSE_VALUE -n NAMESPACE -name topCRName -storageclass_apic STORAGE_CLASS_NAME -operator_in_specific_namespace -production -profile n3xc16.m48
```

- API Connect in Cloud Pak for Integration:
  - If operators are in all namespaces:

```
python3 install_apic_on_ocp.py -license LICENSE_VALUE1 -n NAMESPACE -name topCRName -storageclass_apic STORAGE_CLASS_NAME1 -cp4i -storageclass_pn STORAGE_CLASS_NAME2 -license_pn LICENSE_VALUE2 -production -profile n3xc16.m48
```

- If operators are in a specific namespace:

```
python3 install_apic_on_ocp.py -license LICENSE_VALUE1 -n NAMESPACE -name topCRName -storageclass_apic STORAGE_CLASS_NAME1 -cp4i -storageclass_pn STORAGE_CLASS_NAME2 -license_pn LICENSE_VALUE2 -operator_in_specific_namespace -production -profile n3xc16.m48
```

- If operators are in all namespaces, with additional flags `-production` (for license to be set to production) and `-profile` (using n3 profile in this example):

```
python3 install_apic_on_ocp.py -license LICENSE_VALUE1 -n NAMESPACE -name topCRName -storageclass_apic STORAGE_CLASS_NAME1 -cp4i -storageclass_pn STORAGE_CLASS_NAME2 -license_pn LICENSE_VALUE2 -production -profile n3xc16.m48
```



6. Optional: Install any additional Gateways, Developer Portals, or Analytics subsystems that are needed.
  - Complete this step now if your target system will have additional Gateways, Developer Portals, or Analytics subsystems to match what is there in the source API Connect system, and you want to complete data migration for all subsystems in one pass through the procedures in [Restoring the target system](#).
  - Skip this step now, and proceed to step 7 if either of the following cases are true:
    - If your target system will not need any additional Gateways, Developer Portals, or Analytics subsystems.
    - Your target system will have additional Gateways, Developer Portals, or Analytics subsystems but you want to first complete data migration based on the top CR installation (1 Management, 1 Portal, 1 Gateway, 1 Analytics), before you install any additional subsystems.

Note: To review the data migration choices, see [v10 migration planning](#).

If you choose complete this step now, see:

- [Deploying on OpenShift and Cloud Pak for Integration](#)
  - [Installation procedures with IBM Cloud Pak for Integration](#)
7. Next, restore the configuration from the source system. Continue with [Restoring the target system](#).

---

## Restoring the target system

Restore the saved configuration and backups from the source system onto the target system.

### Before you begin

---

- Complete [Installing the target system](#).
- Ensure that any network route between the target installation and the source system is disabled. The target system must not be able to communicate with the source system.

### About this task

---

Use scripts to:

- Restore the Management subsystem
- Register Gateways and Portals on the target system
- Update Portals in the Catalog to point to new Portal systems
- Restore Portal backups for each site
- Update Gateways in the Catalog to point to new Gateway systems

All scripts run on the target system must be run from a location where the data directory is present so that the source configuration can be used.

Note: You can optionally run scripts in `--silent` mode, particularly for the following scenarios:

- Register Gateways and Portals in the target system. (Step 3)
- Update Portals in the Catalog to point to new Portal systems. (Step 4)
- Update Gateways in the Catalog to point to new Gateway systems. (Step 6)

If there are multiple Portals and Management subsystems present in the data directory (as saved during [Preparing the source system](#)), the restore script prompts you to select the subsystem name to use for migration while running the script. If running in silent mode, that value must be supplied as an additional flag.

Deployments with multiple Portals and Gateways are a common scenario. The saved data can have multiple Management subsystems only if you ran `save_v10_source_configuration.py` against multiple clusters with different Management subsystem names, without cleaning the data directory.

### Procedure

---

1. Before restoring the Management subsystem backup, determine whether there have been any delta changes to your source system since you backed up the Management subsystem database and Portal database.
  - If you shutdown the Postgres database, as instructed in [Installing the target system](#), there are no delta changes. Continue to step 2.
  - If you did not shutdown the Postgres database, there could be some delta changes to the Management database, because the Management subsystem on the source system remains in READY state. In this case, to ensure you capture the latest configuration, repeat [Preparing the source system](#). By doing so, the Management and Portal databases are backed up again. The latest backup will then be used to restore to the target system.
2. Restore a Management subsystem backup onto the target system.
  - a. On the target API Connect system, ensure that the Management subsystem is up and running.
  - b. Run `restore_management_db.py`. Examples:
    - OpenShift or Kubernetes target system:
 

```
python3 restore_management_db.py -n <mgmt namespace>
```
    - or:
 

```
python3 restore_management_db.py -mgmt_ns <mgmt namespace>
```
    - Cloud Pak for Integration target system:
 

```
python3 restore_management_db.py -n <namespace> -s <platform API hostname>
```
    - If for some reason the management subsystem is not healthy, restore using the command:
 

```
python3 restore_management_db.py -n <mgmt namespace> -ignore_health_check
```
3. Register Gateways and Portals on the target system.
  - a. Prerequisites:
    - The `register_gateway_portals_in_target.py` script must be run on the target system after the Management database from the source system has been restored.

- The Gateway and Developer Portal subsystems that need to be registered must be in a healthy state.
- b. Usage notes:
- The credentials provided from this step will be same as the credentials used on the source system, because the Management database from the source system is restored.
  - Keep the endpoints ready for the new Gateway and Portal services so that they can be registered by running `register_gateway_portals_in_target.py`. To view endpoints, use `oc get routes`.
  - If you run `register_gateway_portals_in_target.py` for a Cloud Pak for Integration target system, and you see the error `The API Connect Gateway Service is already registered to another API Manager instance`, restart all the gateway pods, wait for them to become healthy, and run `register_gateway_portals_in_target.py` again.
  - Make sure the mapping of old URLs with the new ones are correct. For example, when the source system is a VMware appliance, if wrongly mapped (wrong values provided) communication errors occur during Gateway registration. Example errors messages:
    - `"Error: An error occurred communicating with the gateways subsystem "`
    - `"error: 'Client network socket disconnected before secure TLS connection was established'"`

On the appliance, the Gateway Manager URL uses port 3000.

c. Run `register_gateway_portals_in_target.py` in either interactive mode or silent mode:

- Interactive mode  
The `register_gateway_portals_in_target.py` script displays the Portal and Gateway endpoints from the source system, and prompts the user to enter the corresponding values from the target system. Examples:

- If Management, Portal and Gateway subsystems are in the same namespace:

```
python3 register_gateway_portals_in_target.py -s <platform API hostname> -u <username> -p <password> -r <realm name> -n <namespace>
```

- If Management, Portal and Gateway subsystems are in different namespaces, use `-mgmt_ns`, `-ptl_ns`, and `-gw_ns` to specify each, for example:

```
python3 register_gateway_portals_in_target.py -s <platform API hostname> -u <username> -p <password> -r <realm name> -mgmt_ns <mgmt namespace> -ptl_ns <portal namespace> -gw_ns <gway namespace>
```

- If you are using SSO authentication, replace:

```
-u <username> -p <password>
```

with:

```
--sso
```

- Multiple namespaces can alternatively be specified with "ns1|ns2|ns3" notation. For example:

```
python3 register_gateway_portals_in_target.py -n "ns1|ns2|ns3" -u admin -p <password> -s <platform API hostname> -r admin/default-idp-1 -silent
```

```
python3 register_gateway_portals_in_target.py -s <platform API hostname> -u admin -p <password> -r admin/default-idp-1 -mgmt_ns ns1 -ptl_ns "ns1|ns2" -gw_ns "ns1|ns2|ns3" -silent
```

- Silent mode

The `gateway_portal_mapping.yaml` file, with mapping between old and new endpoints for Gateway, Portal, and Analytics endpoints, must be located in the same directory as `register_gateway_portals_in_target.py`.

This `gateway_portal_mapping.yaml` file is generated when saving the source API Connect system configuration with properties data filled in. For every Gateway, Portal, and Analytics subsystem registered with Management in the source system, the yaml has the endpoint mapping between source and target systems.

Remaining data must be filled to use the yaml in silent mode. All values must begin with `https://`.

- Example:

```
python3 register_gateway_portals_in_target.py -n <namespace> -u <username> -p <password> -s <platform API hostname> -r <realm name> -silent
```

- Example: If the subsystems are in multiple namespaces.

For example, ns1, ns2, ns3 namespaces contain the subsystems. Both of the following commands work the same way:

```
python3 register_gateway_portals_in_target.py -n "ns1|ns2|ns3" -u <username> -p <password> -s <platform API hostname> -r <realm name> -silent
```

```
python3 register_gateway_portals_in_target.py -s <platform API hostname> -u <username> -p <password> -r <realm name> -mgmt_ns ns1 -ptl_ns "ns1|ns2" -gw_ns "ns1|ns2|ns3" -silent
```

- Example `gateway_portal_mapping.yaml` file with:

- Two Gateways
- One Portal or Analytics subsystem
- All data mapped to the target API Connect system:

```
analytics_mapping:
 registered_analytics_name:
 https://ac.source-apic.example.com: https://a1-a7s-ac-endpoint-apic1.apps.vlxocp-3278.cp.fyre.example.com
gateway_mapping:
 registered_apigw_name:
 https://rgw.source-apic.fyre.example.com: https://a1-gw-gateway-apic1.apps.vlxocp-3278.cp.fyre.example.com
```

```

 https://rgwd.source-apic.fyre.example.com: https://a1-gw-gateway-manager-apic1.apps.vlxocp-
3278.cp.fyre.example.com
 registered_v5gw_name:
 https://gw.source-apic.fyre.example.com: https://v5gw.gw.apps.vlxocp-3278.cp.fyre.example.com
 https://gwd.source-apic.fyre.example.com: https://v5gw.gwd.apps.vlxocp-3278.cp.fyre.example.com
portal_mapping:
 registered_portal_name:
 https://api.portal.source-apic.fyre.example.com: https://a1-ptl-portal-director-apic1.apps.vlxocp-
3278.cp.fyre.example.com
 https://portal.source-apic.fyre.example.com: https://a1-ptl-portal-web-apic1.apps.vlxocp-
3278.cp.fyre.example.com

```

- Example gateway\_portal\_mapping.yaml file with:

- Only two Gateways
- One Portal or Analytics
- One Gateway not mapped

Note that "not mapped" means that the data in the Management subsystem related to this Gateway is not mapped, and the Gateway will not be migrated during this migration phase.

The Gateway could be migrated later; for example, after the second Gateway is created on the target system. Until the data related to the not-migrated Gateway or Portal is migrated or deleted, there will be stale data in the database.

The same behavior can be achieved by leaving the values empty, or removing the entire section (name and mapping).

```

analytics_mapping:
 registered_analytics_name:
 https://ac.source-apic.example.com: https://a1-a7s-ac-endpoint-apic1.apps.vlxocp-
3278.cp.fyre.example.com
gateway_mapping:
 registered_apigw_name:
 https://rgw.source-apic.fyre.example.com: https://a1-gw-gateway-apic1.apps.vlxocp-
3278.cp.fyre.example.com
 https://rgwd.source-apic.fyre.example.com: https://a1-gw-gateway-manager-apic1.apps.vlxocp-
3278.cp.fyre.example.com
 registered_v5gw_name:
 https://gw.source-apic.fyre.example.com: NEW_GATEWAY_API_ENDPOINT_BASE_IN_TARGET_SYSTEM
 https://gwd.source-apic.fyre.example.com: NEW_GATEWAY_ENDPOINT_IN_TARGET_SYSTEM
portal_mapping:
 registered_portal_name:
 https://api.portal.source-apic.fyre.example.com: https://a1-ptl-portal-director-apic1.apps.vlxocp-
3278.cp.fyre.example.com
 https://portal.source-apic.fyre.example.com: https://a1-ptl-portal-web-apic1.apps.vlxocp-
3278.cp.fyre.example.com

```

4. Use `update_to_new_portals.py` to update Portals in the catalog to point to new Portal systems.

Note: If there is no Portal subsystem on the source system, skip this step. Also, skip this step if no Portal sites are created in any of the provider orgs. Continue with [6](#).

- a. Review prerequisites:

- The Management and Portal subsystems on the target API Connect system must be in a healthy state.
- The `update_to_new_portals.py` script must be run on the target system after the following actions have been completed:
  - The Management database from the source system has been restored.
  - The new Gateway or Portal instance is registered in the Cloud Manager, using script or from the UI.
- The credentials for accessing the admin org (Cloud Manager UI) and, if required, the provider org (API Manager UI) must be available for running the script.

- b. Run the `update_to_new_portals.py` script in one of two modes, based on how the Provide Org credentials are available. The modes are:

- Using a common migration user created by the script
- Using credentials provided by the user in a yaml file

Important: The provider org credentials must have administrative access.

Examples:

- Using a common migration user created by the script :

The `update_to_new_portals.py` script runs in this mode by default. This script creates a migration user `muser1` in `migrationur` local user registry, and uses it to migrate Portal information in all the provider orgs. Once the migration is done, the user and user registry are deleted. There is no change in data once the migration is complete.

Examples:

- If the Management and Portal subsystems are in the same namespace

```
python3 update_to_new_portals.py -s <platform API hostname> -u <username> -p <password> -r <realm name> -n
<namespace> -api_manager_hostname <platform API hostname> -silent
```

- If using SSO authentication:

```
python3 update_to_new_portals.py -s <platform API hostname> --sso -n <namespace> -api_manager_hostname
API_MANAGER_HOSTNAME -silent
```

- If the Management and Portal subsystems are in different namespaces:

```
python3 update_to_new_portals.py -s <platform API hostname> -u <username> -p <password> -r <realm name> -mgmt_ns
<mgmt namespace> -ptl_ns <portal namespace>E -api_manager_hostname <platform API hostname>E -silent
```

- Using credentials provided by the user in a `provider_org_credentials.yaml` file:

The `provider_org_credentials.yaml` file, with some data filled in, is generated by the `save_v10_source_configuration.py` script you ran in [Preparing the source system](#). The yaml file is available in the same directory as the script. The script has to be edited and all details filled in before it can be used.

Examples:

- If Management and Portal subsystems are in the same namespace:

```
python3 update_to_new_portals.py -s <platform API hostname> -u <username> -p <password> -r <realm name> -n <namespace> -no_migration_user -silent
```

- If using SSO authentication:

```
python3 update_to_new_portals.py -s <platform API hostname> --sso -n <namespace> -no_migration_user -silent
```

- If Management and Portal subsystems are in different namespaces:

```
python3 update_to_new_portals.py -s <platform API hostname> -u <username> -p <password> -r <realm name> -mgmt_ns <mgmt namespace> -ptl_ns <portal namespace> -no_migration_user -silent
```

Example provider\_org\_credentials.yaml:

- With credentials given for every provider org. In this example, only two orgs are present.

```
provider_org_credentials:
 apiManagerHostName: <platform API hostname>
 PORG_NAME_1:
 password: PASSWORD_FOR_THIS_REALM_TO_BE_CHANGED
 realm: provider/default-idp-2
 username: USERNAME_FOR_THIS_REALM_TO_BE_CHANGED
 PORG_NAME_2:
 password: PASSWORD_FOR_THIS_REALM_TO_BE_CHANGED
 realm: provider/default-idp-2
 username: USERNAME_FOR_THIS_REALM_TO_BE_CHANGED
 useSameCredentialsForAllProviderOrgs: false
```

- With credentials given for just one provider org. Also requires `useSameCredentialsForAllProviderOrgs` set to true. This method can be used only when the same credentials are used for all provider orgs.

Note: In some cases, there are multiple provider orgs in the Management database.

```
provider_org_credentials:
 apiManagerHostName: <platform API hostname>
 PORG_NAME_1:
 password: PASSWORD_FOR_THIS_REALM_TO_BE_CHANGED
 realm: provider/default-idp-2
 username: USERNAME_FOR_THIS_REALM_TO_BE_CHANGED
 useSameCredentialsForAllProviderOrgs: true
```

- With SSO credentials and just one provider org:

```
provider_org_credentials:
 apiManagerHostName: <platform API hostname>
 useSameCredentialsForAllProviderOrgs: false
 org_name:
 #sso authentication
 #the key (that can expire in sometime) has to be generated separately using
 https://<apiManagerHostName>/auth/manager/sign-in/?from=TOOLKIT
 apiKey: some_key
```

## 5. Restore Portal backups for each site.

Note: If there is no Portal subsystem on the source system, skip this step. Continue with [6](#).

To restore the Portal sites on the target system, you can either use the using script `restore_portal_db.py` or complete a manual restore.

- Restoring through the script:

```
python3 restore_portal_db.py -n PORTAL_SUBSYSTEM_NAMESPACE
```

- Restoring through manual steps:

- Log in to the Portal admin container:

```
oc exec -it <www_pod> -c admin -- bash
```

- List the site backups:

```
remote_transfer_backup -l
```

The list of backups that need to be restored is available in the output logs as a report of [Step 4](#).

- Download each site backup, and run it for all sites:

```
remote_transfer_backup -d <SITE_BACKUP_FILENAME>
```

- Restore each of the site backups:

```
restore_site -f -u <SITE_URL> <SITE_BACKUP_FILENAME>
```

Note: If there are more than one Portal subsystems on the target system, as needed to correspond to the Portal subsystems on the source system, run the script for each of the matching source and target Portal subsystems.

For example, if there are two Portals on the target system, matching what is on the source system, and these Portals are installed in namespaces `target1` (for the top-level CR) and `pt12` (for second portal), then the script will be run twice, as follows:

```
python3 restore_portal_db.py -n target1
```

```
python3 restore_portal_db.py -n pt12
```

## 6. Run `update_to_new_gateways.py` to update Gateways in the Catalog to point to new Gateway systems.

- Review prerequisites:

- The Gateway subsystem must be in a healthy state.
- The `update_to_new_gateways.py` script must be run on the target system after the following actions have been completed:
  - The Management database from the source system has been restored.
  - The new Gateway instance is registered in the Cloud Manager, using script or from the UI.

- The credentials for accessing the admin org (Cloud Manager UI) and, if required, the provider org (API Manager UI) must be available for running the script.
- b. Run the `update_to_new_gateways.py` script in one of two modes, based on how the Provider Org credentials are available. The modes are:
- Using a common migration user created by the script
  - Using credentials provided by the user in a yaml file
- Important: The provider org credentials must have administrative access.
- Using a common migration user created by the script:  
The `update_to_new_gateways.py` script runs in this mode by default. This script creates a migration user `muser1` in `migrationur` local user registry, and uses it to migrate Portal information in all the provider orgs. Once the migration is done, the user and user registry are deleted. There is no change in data once the migration is complete.

Examples:

- If the Management and Gateway subsystems are in the same namespace:

```
python3 update_to_new_gateways.py -s <platform API hostname> -u <username> -p <password> -r <realm name> -n <namespace> -api_manager_hostname <platform API hostname> -silent
```

- If using SSO authentication:

```
python3 update_to_new_gateways.py -s <platform API hostname> --sso -n <namespace> -api_manager_hostname <platform API hostname> -silent
```

- If the Management and Gateway subsystems are in different namespaces:

```
python3 update_to_new_gateways.py -s <platform API hostname> -u <username> -p <password> -r <realm name> -mgmt_ns <mgmt namespace> -ptl_ns <portal namespace> -api_manager_hostname <platform API hostname> -silent
```

- Using credentials provided by the user in a `provider_org_credentials.yaml` file:

The `provider_org_credentials.yaml` file, with some data filled in, is generated by the `save_v10_source_configuration.py` script you ran in [Preparing the source system](#). The yaml file is available in the same directory as the script. The script has to be edited and all details filled in before it can be used.

- If the Management and Gateway subsystems are in the same namespace:

```
python3 update_to_new_gateways.py -s <platform API hostname> -u <username> -p <password> -r <realm name> -n <namespace> -no_migration_user -silent
```

- If using SSO authentication:

```
python3 update_to_new_gateways.py -s <platform API hostname> --sso -n <namespace> -no_migration_user -silent
```

- If the Management and Gateway subsystems are in different namespaces:

Note that if multiple gateway subsystems present across namespaces, you can give multiple namespaces separated by the "|" symbol for `-gw_ns`.

```
python3 update_to_new_gateways.py -s <platform API hostname> -u <username> -p <password> -r <realm name> -mgmt_ns <mgmt namespace> -gw_ns GW_NAMESPACE -no_migration_user -silent
```

- Example `provider_org_credentials.yaml`

- With credentials given for every provider org, and only two orgs present:

```
provider_org_credentials:
 apiManagerHostName: <platform API hostname>
 PORG_NAME 1:
 password: PASSWORD_FOR_THIS_REALM_TO_BE_CHANGED
 realm: provider/default-idp-2
 username: USERNAME_FOR_THIS_REALM_TO_BE_CHANGED
 PORG_NAME 2:
 password: PASSWORD_FOR_THIS_REALM_TO_BE_CHANGED
 realm: provider/default-idp-2
 username: USERNAME_FOR_THIS_REALM_TO_BE_CHANGED
 useSameCredentialsForAllProviderOrgs: false
```

- With credentials given for just one provider org, and `useSameCredentialsForAllProviderOrgs` is set to `true`. This can be used only when same credentials are used for all provider orgs.

Note: There can be multiple provider orgs in the Management database.

```
provider_org_credentials:
 apiManagerHostName: <platform API hostname>
 PORG_NAME 1:
 password: PASSWORD_FOR_THIS_REALM_TO_BE_CHANGED
 realm: provider/default-idp-2
 username: USERNAME_FOR_THIS_REALM_TO_BE_CHANGED
 useSameCredentialsForAllProviderOrgs: true
```

- With SSO credentials and just one provider org:

```
provider_org_credentials:
 apiManagerHostName: <platform API hostname>
 useSameCredentialsForAllProviderOrgs: false
 porg_name:
 #sso authentication
 #the key(that can expire in sometime) has to be generated separately using
 https://<apiManagerHostName>/auth/manager/sign-in/?from=TOOLKIT
 apiKey: some_key
```

7. Continue to [Verifying the target system](#).

---

## Verifying the target system

Verify that the target system is in a healthy state and fully configured.

### Before you begin

---

Complete [Restoring the target system](#).

### About this task

---

Check the health of the target system, complete any optional configuration, and validate data, APIs, and URLs.

### Procedure

---

1. Use the `health_check_post_migration.py` script to health check the target system.

Examples:

- When all the subsystems are in the same namespace, and the deployments uses username/password authentication:

```
python3 health_check_post_migration.py -s PLATFORM_API_HOSTNAME -r REALM_NAME -u USERNAME -p PASSWORD -n NAMESPACE
```

- When all the subsystems are in the same namespace, and the deployments uses username/password authentication, with use of the optional flag `-check_portal_site_health` to check the health of Portal sites:

```
python3 health_check_post_migration.py -s PLATFORM_API_HOSTNAME -r REALM_NAME -u USERNAME -p PASSWORD -n NAMESPACE -check_portal_site_health
```

- When subsystems are present in different namespaces, and the deployment uses username/password authentication

```
python3 health_check_post_migration.py -s PLATFORM_API_HOSTNAME -r REALM_NAME -u USERNAME -p PASSWORD -mgmt_ns NAMESPACE -gw_ns NAMESPACE
```

- When all subsystems are in different namespaces, and the deployment uses username/password authentication:

```
python3 health_check_post_migration.py -s PLATFORM_API_HOSTNAME -r REALM_NAME -u USERNAME -p PASSWORD -mgmt_ns NAMESPACE -gw_ns NAMESPACE -ptl_s NAMESPACE -a7s_ns NAMESPACE
```

- When all subsystems are in the same namespace, and the deployment uses SSO authentication:

```
python3 health_check_post_migration.py -n NAMESPACE --sso -s PLATFORM_API_HOSTNAME
```

- When all subsystems are in the same namespace, and the deployment uses SSO authentication silent mode:

```
python3 health_check_post_migration.py -n NAMESPACE --sso -s PLATFORM_API_HOSTNAME -api_key API_KEY_VALUE
```

- When there are multiple subsystems across namespaces:

```
python3 health_check_post_migration.py -s PLATFORM_API_HOSTNAME -r REALM_NAME -u USERNAME -p PASSWORD -n "namespace1|namespace2|namespace3"
```

2. Complete any optional configuration steps needed for your deployment:

- a. If necessary, update redirect URLs in the authorization server.

If OIDC or SSO authentication is being used, the redirect URLs must be updated in the authorization server after migration. For example, for Google OIDC, you must to add the new redirect URLs in Google Developer Console. The new redirect URLs can be found in the user registry in Cloud Manager.

- b. If necessary, complete configuration for Analytics.

If the Analytics service is present on the target system, backups must be configured. Also, use the Cloud Manager UI to manually associate the Analytics with the Gateways, as needed.

- c. If the target system is Cloud Pak for Integration, you must use the password from the target API Connect system to log in to Platform Navigator.

- d. If necessary, update Portal and Gateway services names in scripts or utilities.

The migrated Portal and Gateway services will have new names (old\_name + "new"). Any scripts and utilities written using the apic toolkit must be updated if they are using names as input parameter and not the ids. This is applicable to only Portal and Gateway names.

3. Validate configuration data:

- a. Validate the data in the target API Connect system by logging into Cloud Manager and API Manager UIs.

- b. Validate the APIs published in the Gateways.

- c. Validate the Portal site URLs

- d. If necessary, update the DNS mapping info.

Once all of the APIs are verified, you may need to update the DNS mapping info, to redirect the calls to the new Gateways and Portals.

The v10 migration is now complete.

4. After migration is finished, and verification of the target system is complete, you can uninstall API Connect on the source system.

See [Uninstalling API Connect](#).

---

## Migrating from OpenShift to Cloud Pak for Integration on the same cluster

Migrate your API Connect v10 deployment from OpenShift to Cloud Pak for Integration, while remaining on the same cluster.

## Before you begin

Your API Connect instance must use the top-level CR. Direct migration of individual API Connect subsystem installations to Cloud Pak for Integration by the steps described here is not supported. Alternative methods of migration if you are not using the top-level CR are:

- If the same subsystem endpoints can be used, deploy a new top-level CR API Connect and restore disaster recovery backups from your existing installation, see [Disaster recovery on OpenShift and Cloud Pak for Integration](#). The new top-level CR deployment can then be migrated to Cloud Pak for Integration following the steps here.
- If different subsystem endpoints need to be used, migrate your API Connect deployment to a new top-level CR OpenShift cluster first. Follow the form factor migration steps [Migrating from v10 to v10 on a different form factor](#), and then return here to complete the OpenShift top-level CR to Cloud Pak for Integration migration.

## About this task

Migration from OpenShift to Cloud Pak for Integration on the same cluster results in changes to the Management subsystem API endpoint URIs, as shown in these examples.

- apiManager endpoint changes from  
`https://topcr-mgmt-api-manager-apic.example.com/manager`  
to  
`https://cpd-apic.example.com/integration/apis/apic/topcr/manager`
- cloudManager endpoint changes from  
`https://topcr-mgmt-admin-apic.example.com/admin`  
to  
`https://cpd-apic.example.com/integration/apis/apic/topcr/admin`
- platformAPI endpoint changes from  
`https://topcr-mgmt-platform-api-apic.example.com/`  
to  
`https://cpd-apic.example.com/integration/apis/apic/topcr/api`
- consumerAPI endpoint changes from  
`https://topcr-mgmt-consumer-api-apic.example.com/`  
to  
`https://cpd-apic.example.com/integration/apis/apic/topcr/consumer-api`

## Procedure

1. Update your `<TOP_CR_NAME>-mgmt-admin-pass` secret with your current Cloud Manager admin password:
  - a. Check whether the `<TOP_CR_NAME>-mgmt-admin-pass` password already matches your Cloud Manager admin password with:

```
oc get secret -n <APIC-namespace> | grep mgmt-admin-pass
```

```
oc get secret -n <APIC-namespace> <secret_name_from_previous_command> -o jsonpath="{.data.password}" | base64 -d && echo ""
```

If the passwords match then skip to [2](#).

- b. From where you run OpenShift CLI commands, use the base64 command to encode your Cloud Manager admin password.

```
echo -n '<password>' | base64
```

Note: If the `base64` command is not available, alternative commands or tools can be used.

- c. Edit the `<TOP_CR_NAME>-mgmt-admin-pass` secret:

```
oc edit secret <TOP_CR_NAME>-mgmt-admin-pass -n <namespace>
```

- d. Replace the `data.password` value with your base64 encoded value from [1.b](#).

```
...
data:
 email: ZXhhbXBsZUBleGFtcGxlLmNvbQ==
 password: <base64 encoded password>
kind: Secret
metadata:
 ...
```

Note: The password in `<TOP_CR_NAME>-mgmt-admin-pass` is only used on API Connect installation, form factor migration, and disaster recovery. It does not need to be kept in sync with the Cloud Manager during normal API Connect usage.

2. Install IBM Cloud Pak Platform UI and create an instance of it.
3. Restart the `apiconnect` operator pod.  
The restarted operator pod detects Cloud Pak for Integration and reconciles the subsystems with it. Wait for the subsystems to be healthy.
4. Log in to Cloud Manager and check the services under the topology section.  
**Known issue:** If you see an additional Analytics service with a different name for the same Analytics endpoint, you must delete the newly created Analytics service.

5. Check that the configurator job is complete: :

```
oc get pods -n <namespace>
...
NAME READY STATUS RESTARTS AGE
<instance name>-configurator-<random string> 0/1 Completed 0 2h
...
```

Complete the following validation checks:

- Verify you are able to login to the Cloud Pak for Integration Platform UI, and can see your API Connect instance.
- Verify you are able to login to Cloud Manager UI, API Manager UI, and your portal sites.
- Test that your APIs are running, and analytics data is logged.
- Test lifecycle operations such as a product publish or delete, or an app creation.

---

## v10 migration troubleshooting

Troubleshoot problems with v10 migration.

Topics:

- [Script help](#)
- [Restarting migration](#)
- [Factory reset of portal subsystem](#)
- [Avoid error when a gateway is already registered to another API manager](#)

---

## Script help

```
python3 <SCRIPT_FILE_NAME> --help
```

---

## Restarting migration

On the target system, after running `register_gateway_portals_in_target.py` or `update_to_new_portals.py` or `update_to_new_gateways.py` script:

- If you want to restore the Management database and try the steps again:
  - Run the restore Management db (Step 2 on [Restoring the target system](#)) with the `-reset_gateway_portal` flag as shown

```
python3 restore_management_db.py -n namespace -s PLATFORM_API_HOSTNAME -reset_gateway_portal
```
  - If running on a Kubernetes environment, supply the Cloud Manager admin password by using the `-p` flag. After this, continue with next steps of migration.
- If you deleted the data directory after registering Gateways or Portals, but want to use the same target API Connect system which you already installed, then:
  1. Complete [Preparing the source system](#) to create the data directory.
  2. Run the restore management db (Step 2 on [Restoring the target system](#)) with the `-reset_gateway_portal` flag as shown

```
python3 restore_management_db.py -n namespace -s PLATFORM_API_HOSTNAME -reset_gateway_portal
```
  - If running on a Kubernetes environment, supply the Cloud Manager admin password by using the `-p` flag. After this, continue with next steps of migration.

---

## Factory reset of portal subsystem

You can do a factory reset of the Portal subsystem by either manually or by using a migration script.

- Using migration script  
The script will restore the Management database and will factory reset the Portal subsystem, and also delete Gateway pods. The gateway pods will get restarted.  
The script will factory reset the Portal subsystem and also delete Gateway pods. When Gateway pods are deleted, they are also restarted.

```
python3 restore_management_db.py -n namespace -s PLATFORM_API_HOSTNAME -reset_gateway_portal
```

  
If using a Kubernetes environment, you must also supply the Cloud Manager admin password by using the `-p` flag. After this command, you must run the remaining steps of migration, starting with Step 3 in [Restoring the target system](#).

```
python3 register_gateway_portals_in_target.py -n NAMESPACE -u USERNAME -p PASSWORD -s HOSTNAME -r REALM_NAME -reset_gateway_portal
```
- Manual steps
  1. Log in to the Portal www pod:

```
kubect1 exec -it PORTAL_WWW_POD_NAME -c admin -n NAMESPACE -- bash
```
  2. Run the command :

```
curl -X DELETE -k --cert /etc/nginx/ssl/portal-director-client-both.pem https://localhost:3009/v2/portal/factory-reset
```
  3. If any sites are there, use the `list_sites` command to check if all sites are deleted.
  4. Check health of the Portal subsystem.

---

## Avoid error when a gateway is already registered to another API manager

Error messages



Error: An error occurred communicating with the gateways subsystem at 'some\_url' (status: 400, response: '"Unable to perform initial registration for API Management.  
Error: The API Connect Gateway Service is already registered to another API Manager instance"

You can fix the error condition manually, or use a script:

- Manual step, when Gateway is in a Kubernetes or OpenShift cluster:  
Restart all the gateway pods.
- Using the migration script when Gateway is in a Kubernetes or OpenShift cluster:  
The script will restore the Management database, factory reset the Portal subsystem, and delete the Gateway pods. Once deleted, Gateway pods get restarted.
  1. Run the following script:

```
python3 restore_management_db.py -n namespace -s PLATFORM_API_HOSTNAME -reset_gateway_portal
```

If using a Kubernetes environment, you must also supply the Cloud Manager admin password by using the `-p` flag.
  2. After this command, you must run the remaining steps of migration, starting with Step 3 in [Restoring the target system](#).
- Gateway is on an Appliance  
Reboot all the Gateway appliances.

---

## Shutting down the Postgres database

Shut down the Postgres database to ensure no changes are made to the database during migration.

Instructions:

- [Shutting down Postgres database in Management subsystem on Kubernetes or VMware](#)
- [Shutting down Postgres database in Management subsystem on OpenShift](#)

---

## Shutting down Postgres database in Management subsystem on Kubernetes or VMware

When using s3 backup configuration for the Management subsystem, the following steps should be run on the source system, before the target API Connect system is installed.

1. If on a VMware appliance deployment, connect to the Management node and log in as root user (`sudo -i`), so that you can run `kubect1`.
2. Enable pgo client:

```
kubect1 set env deploy/ibm-apiconnect ENABLE_PGO_CLIENT="true" -n OPERATOR_NAMESPACE
```

3. Wait for `pgo-client` pod to come up.
4. Bring down the apic operator:

```
kubect1 scale deploy ibm-apiconnect --replicas=0 -n OPERATOR_NAMESPACE
```

5. Scale down deployments.  
Before scaling down the deployments, make a note of the current replicas:

```
kubect1 get deployment -n namespace
```

- Standard deployments :
    - juhu
    - portal-proxy
    - apim
    - lur
    - taskmanager
    - hub (when Test and Monitor is enabled)
- a. Scale down:

```
kubect1 scale deploy <deployment_name> --replicas=0 -n NAMESPACE
```

- b. To get the name of the Postgres cluster:

```
kubect1 get pgcluster -n NAMESPACE
```

- c. Connect to pgo client pod and run:

```
pgo update cluster PG_CLUSTER_NAME --shutdown -n NAMESPACE
```

---

## Shutting down Postgres database in Management subsystem on OpenShift

When using s3 configuration, the following steps should be run on the source system, before the target API Connect system is installed.

1. Change to operator namespace:

```
oc project "{OPERATOR_NS}"
```

2. `oc edit csv ibm-apiconnect.v2.4.0`

- a. Search for `APPLIANCE` and add these two lines above it:

```
- name: ENABLE_PGO_CLIENT
 value: "true"
```

- b. Save and wait for the `pgo-client` pod to come up.

3. `oc scale deploy ibm-apiconnect --replicas=0`
4. Bring down apiconnect operator pod:
 

```
oc scale deploy ibm-apiconnect --replicas=0
```
5. Change to the namespace where API Connect is installed:
 

```
oc project "{APIC NS}"
```
6. Scale down the following deployments: juhu, portal-proxy, apim, lur, taskmanager
 

```
oc scale deploy JUHU_DEPLOYMENT_NAME --replicas=0
oc scale deploy PORTAL_PROXY_DEPLOYMENT_NAME --replicas=0
oc scale deploy APIM_DEPLOYMENT_NAME --replicas=0
oc scale deploy LUR_DEPLOYMENT_NAME --replicas=0
oc scale deploy TASKMANAGER_DEPLOYMENT_NAME --replicas=0
```
7. To get the name of the pg cluster:
 

```
oc get pgcluster
```
8. Log in to pgo client pod:
 

```
oc exec -it $(oc get po|grep pgo|awk '{print $1}') -- bash
```
9. Shutdown Postgres database:
 

```
pgo update cluster PG_CLUSTER_NAME --shutdown -n NAMESPACE
```

## Starting the Postgres database in the Management subsystem on Kubernetes and OVA

Use these steps when you are moving back to the source system because the migration failed, and you want to bring up the Postgres database, if it was shutdown.

1. If on an appliance (VMware), connect to the Management node and log in as root user (`sudo -i`), so that you can run the `kubectl` commands.
2. Get the name of the Postgres cluster:
 

```
kubectl get pgcluster -n NAMESPACE
```
3. Connect to the pgo client pod and run the command:
 

```
pgo update cluster PG_CLUSTER_NAME --startup -n NAMESPACE
```
4. Scale up the deployments that use Postgres.  
Standard deployments : juhu, portal-proxy, apim, lur, taskmanager, hub (when test and monitor is enabled)
 

```
kubectl scale deploy DEPLOYMENT_NAME --replicas=N -n NAMESPACE
```

The value of **N** is from the `kubectl get deployment -n namespace` output that you got before shutting down the Postgres database. It is 1 if using n1 profile, and 3 when using n3 profile.
5. Bring up the apic operator:
 

```
kubectl scale deploy ibm-apiconnect --replicas=1 -n OPERATOR_NAMESPACE
```
6. If s3 backups are used, perform a Management restore with the backup that was taken as part of [Preparing the source system](#).  
Note: Any backups that are taken on the target system are not valid for the source system. The backups must not be used during restoration.
7. Ensure that all subsystems are healthy.

## Starting postgres database on the Management subsystem on OpenShift

Use these steps when you are moving back to the source system because the migration failed, and you want to bring up the Postgres database, if it was shutdown.

1. Change to the project where the top-level CR (apic) is installed:
 

```
oc project APIC_NAMESPACE
```
2. Get the name of the Postgres cluster:
 

```
oc get pgcluster
```
3. Log in to pgo client pod:
 

```
oc exec -it $(oc get po|grep pgo|awk '{print $1}') -- bash
```
4. Start the postgres database:
 

```
pgo update cluster PG_CLUSTER_NAME --startup -n NAMESPACE
```
5. Scale up the deployments: juhu, portal-proxy, apim, lur, and taskmanager. Use the following commands, where **N** is the value of replicas. If an n1 profile being used the value is 1, and if an n3 profile is being used the value is 3.
  - `oc scale deploy JUHU_DEPLOYMENT_NAME --replicas=N`
  - `oc scale deploy PORTAL_PROXY_DEPLOYMENT_NAME --replicas=N`
  - `oc scale deploy APIM_DEPLOYMENT_NAME --replicas=N`
  - `oc scale deploy LUR_DEPLOYMENT_NAME --replicas=N`
  - `oc scale deploy TASKMANAGER_DEPLOYMENT_NAME --replicas=N`
6. Bring down the apiconnect operator pod:
 

```
oc scale deploy ibm-apiconnect --replicas=1 -n OPERATOR_NAMESPACE
```
7. If s3 backups are used, perform a Management restore with the backup that was taken as part of [Preparing the source system](#).

Note: Any backups that are taken on the target system are not valid for the source system. The backups must not be used during restoration.  
 8. Ensure that all subsystems are healthy.

## Migrating from v2018 to v10 on a different form factor

Migrate an API Connect v2018 deployment to a different form factor on v10. For example, migrate v2018 on Kubernetes to v10 on OpenShift.

Use the v2018 to v10 form factor migration process in the following circumstances:

- You want to change the deployment form factor, for example, moving from v2018 on OVA to v10 on OpenShift.
- You are on an old version of API Connect and are unable or prefer not to upgrade to the latest v2018 fix pack to do in-place upgrade. Form factor migration to v10 is supported for v2018 releases back to v2018.4.1 Fixpack 13.
- You are unable to upgrade your v2018 platform (OpenShift or Kubernetes) source environment to a version that supports v10.
- You want to change your API Connect endpoint URLs, but want to stay on the same form factor, for example, changing the portal web endpoint from `https://portal.v2018inst.example.com` to `https://portal.v10inst.example.com`.

If you do not want to change your deployment form factor and do not want to change your endpoint URLs, then the in-place upgrade is the simplest method to move to v10. See the upgrade documentation corresponding to your platform:

- [Upgrading from 2018 on native Kubernetes.](#)
- [Upgrading from 2018 on OpenShift.](#)
- [Upgrading from 2018 on VMware.](#)

Complete the following tasks to migrate your v2018 API Connect deployment to a different form factor on v10:

- [Migration overview](#)  
An overview of the migration steps, the supported paths, and the initial planning for a v2018 to v10 form factor migration.
- [Migration preparation](#)  
Ensure that you have all the information, scripts, and tools ready to complete the migration.
- [v2018 migration steps](#)  
Extract v2018 source data, install v10, load source data into v10 installation.
- [Troubleshooting the form-factor migration from v2018](#)  
Troubleshoot problems with migrating from v2018 to v10 on a different form factor.

## Migration overview

An overview of the migration steps, the supported paths, and the initial planning for a v2018 to v10 form factor migration.

## What does form factor migration involve?

- The health of the source v2018 deployment is verified, and any necessary cleanup of the database is completed.
- Scripts are run against the source deployment to extract Management and Portal data.
- The v10 target deployment is installed.
- A mapping file of the API Connect endpoints is generated, mapping your v2018 endpoint URLs to the endpoints of the v10 deployment.
- Scripts are run against the target deployment to load the data from the v2018 source deployment, applying the endpoint mappings.

The scripts that are used for the migration are in Python and are supported on Linux and MacOS. On Kubernetes and OpenShift the scripts must be run from a client computer that has access to your Kubernetes or OpenShift cluster. On OVA deployments, they are run directly on the Management and Portal VMs.

## Supported v2018 to v10 migration paths

The supported migration paths cover two use cases:

- Migrating from a v2018 system on one form factor to a v10 system on another form factor.
- Migrating from a v2018 system on one form factor to a v10 system on the same form factor, when you want to change the API Connect endpoint URLs.

Table 1. Supported migration paths to API Connect v10

| Source deployment platform of API Connect v2018 | Target deployment platform, based on API Connect v10                                       |
|-------------------------------------------------|--------------------------------------------------------------------------------------------|
| Kubernetes                                      | Cloud Pak for Integration                                                                  |
| Kubernetes                                      | OpenShift Container Platform                                                               |
| Kubernetes                                      | Kubernetes, when moving to a different cluster, resulting in hostname changes              |
| VMware                                          | Cloud Pak for Integration                                                                  |
| VMware                                          | OpenShift Container Platform                                                               |
| VMware                                          | VMware                                                                                     |
| OpenShift Container Platform                    | Cloud Pak for Integration, moving to a different cluster, resulting in hostname changes    |
| OpenShift Container Platform                    | OpenShift Container Platform, moving to a different cluster, resulting in hostname changes |
| Cloud Pak for Integration                       | Cloud Pak for Integration, moving to a different cluster, resulting in hostname changes    |

## v10 migration planning

---

Points for consideration in v2018 to v10 migration planning:

Note: The v2018 deployment is referred to as the 'source' system, and the v10 deployment you are migrating to is referred to as the 'target' system.

- Do you want to migrate all your v2018 source data?  
You do not have to migrate all your v2018 Portal and Gateway subsystems. You can start with a v10 target deployment that has one gateway and portal, and migrate your v2018 data to it. Later you can then add more gateway and portal subsystems to your v10 deployment and migrate your remaining v2018 subsystems. This method is called a staged migration.  
  
Other v2018 data such as organizations, products, and APIs that you do not want can be deleted from your v10 Cloud Manager and the API Manager UIs.
- Delta changes  
The first step in the migration is extracting the data from your v2018 source deployment. If the v2018 deployment continues in use after the data extraction, then any new changes will not be migrated. You must manually apply any delta changes to your v10 target deployment after the v2018 extracted data is restored.
- Sizing  
Ensure that v10 capacity planning is done. Ensure you have sufficient memory, CPU, and disk size, to meet your requirements. See [Planning your deployment](#).
- Time to complete migration.  
The time taken to complete migration depends on the size of your Management database and number of portal sites.

---

## Migration preparation

Ensure that you have all the information, scripts, and tools ready to complete the migration.

---

## Decide on your endpoint mappings

Decide on the new endpoints for your v10 deployment. The v2018 endpoints that require mapping for v10 are:

- Management
  - `cloudManagerEndpoint`
  - `apiManagerEndpoint`
  - `platformAPIEndpoint`
  - `consumerAPIEndpoint`
- Portal
  - `portalAdminEndpoint`
  - `portalUIEndpoint`
- Gateway
  - `gatewayEndpoint`
  - `gatewayManagerEndpoint`
- Analytics
  - `ingestion`

You configure your v10 endpoint URLs when you install v10. You must keep a note of these endpoints and add them to a mapping file that is generated by the v2018 extract script.

Note: It is not a requirement that the endpoints change, in which case you do not need to update the mapping file.

---

## v2018 health checks

Verify that your v2018 deployment is running, APIs operational, and all UIs are accessible.

Ensure that you have recent backups of your Management and Portal subsystems.

---

## v2018 source extraction tools and scripts

The following tools and scripts must be available in your v2018 project directory to run the extract script:

- The form factor migration script: `save_v2018_source_configuration.py`.
- The `apicops` tool.
- The toolkit CLI `apic-slim` command, which must be renamed to `apic`.
- The `apicup` tool.

Follow these steps to download the tools and configure your v2018 project directory:

1. The migration scripts and extract docker image are included in the v10 release and can be obtained from [Passport Advantage](#) or from [IBM Fix Central](#).
  - Download `apiconnect-operator-release-files_<version number>.zip`. The migration scripts are in the `helper_files/formFactorMigration` directory.

```
apiconnect-operator-release-files_<version number>/helper_files/formFactorMigration
```

Copy the `save_v2018_source_configuration.py` file contained in this directory to your project directory.

- **OVA users:** Download `v2018_ova_upgrade_files_<version number>.tgz`. The extract docker image is in this compressed file under:

```
build/v2018-dr-upgrade/extract-image.tar.gz
```

Copy the `extract-image.tar.gz` file to your project directory.

Note: Fix Central is intended for users with existing API Connect installations to find updates. For this reason, to find the latest migration scripts on Fix Central select the previous release number in the menu labeled Installed Version\*. For example, if the latest v10 release is v10.0.5.2, select v10.0.5.1 in Fix Central.

- The **apicops** tool is used to verify the health of the v2018 Management subsystem database, see [Installing apicops](#). Ensure you download the latest v2018 version of **apicops**, not the v10 version. Rename the downloaded **apicops-`<os_name>`** file to just **apicops**, copy it to your project directory, and ensure that it has execute permissions.
- The migration scripts use the API Connect toolkit CLI, see [Installing the v2018 toolkit](#). The v2018 toolkit CLI **apic-slim** executable file must be renamed to **apic** and be in your **PATH** variable when you run the **save\_v2018\_source\_configuration.py** script.
- Copy the v2018 **apicup** executable file to your project directory. This **apicup** file is the same as the one that you used to upgrade to your current v2018 release, so it might already be there.

**OVA users:** Copy your project directory to your Management VM and Portal VMs. If you have a three node deployment, copy the project directory to one VM in each cluster. To access your VMs, SSH to your VM as the **apicadm** user and run `'sudo -i'`. Create a directory called **ffm**, copy your project directory there, and add this directory to your **PATH** variable:

```
ssh apicadm@<VM hostname>
sudo -i
mkdir ffm
scp -r <remote project directory location> /root/ffm
export PATH=/root/ffm/<project dir>:$PATH
```

Warning: `/root/ffm/<project dir>` must be set at the front of the **PATH** variable to prevent a different executable file that is present on OVA and also called **apic** from being run.

#### Non-OVA users

- Ensure that Python 3 and the PyYAML python module are present and usable from your project directory. To install the PyYAML module run:

```
pip3 install pyyaml
```

- From your project directory, you must be able to run **kubectl** (Kubernetes) or **oc** (OpenShift) commands to access the source system cluster.
- The user that runs the **save\_v2018\_source\_configuration.py** script must have execute permission on the **apicops**, **apicup**, and **apic** commands.
- The user that runs the script must have write access in the project directory, so the exported data can be saved there.
- The full path of your project directory must not contain any space characters.
- Add the project directory to your **PATH** variable:

```
export PATH=<project dir>:$PATH
```

- Load the v10 install images into your image registry, these include the **ibm-apiconnect-management-v2018-to-v10-extract** image that you need to extract your v2018 source data.

The Management subsystem endpoints must be reachable from the location of your project directory.

When your project directory is set up to run the migration, its contents should look like this:

```
ls -l
total 117424
...
-rwxr-xr-x 1 root root 54347564 Jul 1 08:45 apicops
-rwxr-xr-x 1 root root 43742016 Jan 27 2020 apic
-rwxr-xr-x 1 root root 36999168 Oct 21 13:00 apicup
-rw-r--r-- 1 root root 86878 Oct 21 13:27 save_v2018_source_configuration.py
-rw-r----- 1 root root 10682 Oct 24 10:06 apiconnect-up.yml
drwxr-xr-x 3 root root 4096 Oct 24 12:58 build (OVA ONLY - CONTAINS UNPACKED v2018_ova_upgrade_files_<version
number>.tgz)
...
which apicops
/root/ffm/<project dir>/apicops

which apic
/root/ffm/<project dir>/apic

which apicup
/root/ffm/<project dir>/apicup
```

Note: This example is from an OVA deployment. File ownerships can differ on a non-OVA deployment.

## Credentials and access requirements

- The migration scripts require the Cloud Manager admin user credentials.
- The computer where you have your project directory must have network access to the Management Platform REST API endpoint.

## Backing up v2018 Management and Portal

The migration process checks that backups for Management and Portal subsystems exist. The migration scripts exit if no backups are detected.

- Kubernetes and OpenShift: [https://www.ibm.com/docs/en/SSMNED\\_2018/com.ibm.apic.install.doc/overview\\_backup\\_restore.html](https://www.ibm.com/docs/en/SSMNED_2018/com.ibm.apic.install.doc/overview_backup_restore.html)
- VMware: [https://www.ibm.com/docs/en/SSMNED\\_2018/com.ibm.apic.install.doc/overview\\_backup\\_restore\\_vm.html](https://www.ibm.com/docs/en/SSMNED_2018/com.ibm.apic.install.doc/overview_backup_restore_vm.html)

## Additional requirements and limitations

- If a custom gateway image is being used on the source system, an equivalent v10 based image must be used on the target system.
- Analytics data is not migrated.
- If the target platform is Cloud Pak for Integration, make sure the Cloud Manager password for **admin** user is not **7iron-hide** on the source cluster. If the default password **7iron-hide** is used, migration will fail after step 5 in [Restoring the v2018 data to the v10 target deployment](#), because the configurator job assumes that

this is a new installation, and changes the password automatically.

4. Ensure that any network route between the target installation and the source system is disabled. The target system must not be able to communicate with the source system.

## Final readiness checks

---

Verify that `apicops`, `apic` (renamed from `apic-slim`), `kubect1/oc`, and `apicup` commands all run successfully from your v2018 project directory.

- Verify that the `kubect1` command returns a list of running pods (`oc` command on OpenShift). Example from Management server:

```
kubect1 get pods
NAME READY STATUS RESTARTS AGE
apiconnect-analytics-proxy-8d6c595fb-4z6kc 1/1 Running 0 25h
...
```

- Verify that `apicops` can access your Management database by requesting a list of catalogs:

```
apicops catalogs:list
Current local date/time 10/21/2022, 12:34:24 PM [Current UTC time: 2022-10-21T12:34:24.354Z]
apicops 0.2.215
Pod version is: 2018.4.1.9-ifix1.0 (apim:2018.4.1.9-ifix1.0-15-fffd5fa)
All catalogs in format (org UUID . catalog UUID => org name . catalog name)
3ac560d2-9d99-43fd-ae9f-dfcc0321044a . 803a3ba3-1903-4b64-8c33-47d236bf1546 => testorg . sandbox
...
```

Note: OVA users: `apicops` is only required on your Management VM, not your Portal VM.

- Verify that you can log in to your Management subsystem with the toolkit CLI. For more information, see [v2018 toolkit CLI login](#):

```
apic login --server <management server endpoint> --username admin --realm admin/default-idp-1
Enter your API Connect credentials
Password?
Logged into <management server endpoint> successfully
```

Note: OVA users: You need to run the toolkit CLI on your Management VM only.

- Verify that the `apicup` command returns a list of your subsystems:

```
apicup subsys list
Name Type Target
test-v2018-management ManagementSubsystem appliance
test-v2018-portal PortalSubsystem appliance
test-v2018-analytics AnalyticsSubsystem appliance
```

## v10 target deployment preparation

---

**Non-OVA users:** On a computer that has network access to the v10 environment, create a directory that is called `apicv10install`. Ensure the `apicv10install` directory meets the following requirements:

- Python 3 and the PyYAML python module are present and usable from your `apicv10install` directory. To install the PyYAML module run:

```
pip3 install pyyaml
```

- From your `apicv10install` directory, you must be able to run `kubect1` (Kubernetes) or `oc` (OpenShift) commands to access the target deployment environment.
- The full path of your `apicv10install` directory must not contain any space characters.
- The computer where you have your `apicv10install` directory must have network access to the v10 Management REST API endpoints.

**OVA users:** Create your v10 project directory, as described in [First steps for deploying in a VMware environment](#). You can name the project directory what you want, but this topic assumes it is called `apicv10install`.

**All users:** From the unpacked `apiconnect-operator-release-files_<version number>.zip`, copy the contents of `helper_files/formFactorMigration` to your `apicv10install` directory.

## v2018 migration steps

---

Extract v2018 source data, install v10, load source data into v10 installation.

### Before you begin

---

Review [Migration overview](#) and complete [Migration preparation](#).

- [Extracting the v2018 source data on Kubernetes and OpenShift](#)  
Run the `save_v2018_source_configuration.py` script to gather the source deployment's configuration settings.
- [Extracting the v2018 source data on VMware](#)  
Run the `save_v2018_source_configuration.py` script to gather the source deployment's configuration settings.
- [Installing the v10 target on Kubernetes](#)  
Prepare your Kubernetes environment, create your API Connect secrets, install v10.
- [Installing the v10 target system on OpenShift and Cloud Pak for Integration](#)  
On the target system, configure a cluster, create the necessary secrets, and install API Connect.
- [Installing the v10 target system on VMware](#)  
Install your v10 OVA deployments and prepare them for restoring your v2018 data.
- [Restoring the v2018 data to the v10 target deployment](#)  
Restore the saved configuration and backups from the source deployment onto the target deployment.

- [Verifying the v10 target deployment](#)  
Verify that the target deployment is working correctly. Remove any unwanted v2018 data and subsystems.
- [Migrating additional v2018 subsystems: Staged migration](#)  
If you did not migrate all your v2018 portal and gateway subsystems initially, and now want to migrate the remaining ones.

## Extracting the v2018 source data on Kubernetes and OpenShift

Run the `save_v2018_source_configuration.py` script to gather the source deployment's configuration settings.

### Before you begin

Complete the steps described in [Migration preparation](#).

### About this task

Note: It is recommended that you keep the console output of all python scripts that are run. On most MacOS and Linux environments, appending the following to the end of the command saves the output to the specified file:

```
2>&1 | tee <output filename>
```

### Procedure

1. Verify and fix any inconsistencies in the Management database by running the `apicops preupgrade` command:

```
apicops preupgrade -m fix --upgrade_to v10 -l apicopsOutput.txt
```

This command checks for known data issues and cleans up the data. The time the command takes to run depends on the size of the management database. `apicops` reports any problems that it is unable to fix at the end of the `apicopsOutput.txt` output file. If `apicops` reports any problems, open a support case and include the `apicopsOutput.txt` file, do not attempt the migration until the case is resolved.

2. Extract the v2018 data by running the `save_v2018_source_configuration.py` script.

Note: The extract script must be run within 2 hours of running the `apicops` health check.

On Kubernetes, if all subsystems are in the same namespace, run:

```
python3 save_v2018_source_configuration.py -n <namespace> -u admin -p <password> -s <platform_api_endpoint> -r admin/default-idp-1 --storage_class <storage class> -registry_secret <image_pull_secret> -extract_image <extract_image_url>
```

On OpenShift, if all subsystems are in the same namespace, run:

```
python3 save_v2018_source_configuration.py -ocp -n <namespace> -u admin -p <password> -s <platform_api_endpoint> -r admin/default-idp-1 --storage_class <storage class> -registry_secret <image_pull_secret> -extract_image <extract_image_url>
```

Where

- `<namespace>` is your v2018 deployment's namespace.
- `<password>` is the Cloud Manager UI admin password.
- `<platform_api_endpoint>` is the FQDN of the platform api endpoint that is defined for your v2018 deployment. To find this run:
  - OpenShift:

```
oc get routes -n <apic namespace> | grep platform-api
```

- Kubernetes:

```
kubectl get ingress -n <apic namespace> | grep platform-api
```

Or get this value from the `apiconnect-up.yml` file in your project directory.

- `<storage class>` is the name of the storage class your v2018 API Connect deployment is using. Get this value from the `apiconnect-up.yml` file in your project directory.
- `<image_pull_secret>`: Is the secret used to pull the API Connect images. Get this value from the `apiconnect-up.yml` file in your project directory.
- `<extract_image_url>`: is the URL of the API Connect extract load image `ibm-apiconnect-management-v2018-to-v10-extract`. To find the full URL, list all images in your registry. The extract load image URL looks like this:

```
<registry_server>/ibm-apiconnect-management-v2018-to-v10-extract@sha256:521c4c4a36bf08cd9544084aa69f333f751376c89d0bd51f8e85d27ac00e893f
```

Note: This image is one of the v10 images that you loaded during migration preparation: [Migration preparation](#).

Example (Kubernetes):

```
python3 save_v2018_source_configuration.py -n apicsrc -u admin -p cuISlavSuFImPjwW5cjmMJwLZt -s api.ops-isl.jp-tok.apiconnect.cloud.ibm.com -r admin/default-idp-1 --storage_class apic-mz-ibmc-block-gold-monthly -registry_secret apiconnect-image-pull-secret -extract_image apic-dev-docker-local.artifactory.swg-devops.com/ibm-apiconnect-management-v2018-to-v10-extract@sha256:6ec184f756e03f916b1bb1b22952d8ecc2dff1b445b3c81b9fa943d775ab665a
```

If Management, Portal, and Gateway subsystems are in different namespaces, use `-mgmt_ns`, `-ptl_ns`, and `-gw_ns` to specify each. Example (Kubernetes):

```
python3 save_v2018_source_configuration.py -mgmt_ns <management_namespace> -ptl_ns <portal_namespace> -a7s_ns <analytics_namespace> -gw_ns <gateway_namespace> -u admin -p <password> -s <platform_api_endpoint> -r admin/default-idp-1 --storage_class <storage class> -registry_secret <image_pull_secret> -extract_image <extract_image_url>
```

Note: Command help for the script is available with `--help`:

```
python3 save_v2018_source_configuration.py --help
```

3. The save script creates a `data` directory and two yaml files that are called `gateway_portal_mapping.yaml` and `provider_org_credentials.yaml`. Copy the data directory and the two yaml files to your v10 `apicv10install` directory.

## What to do next

---

Install your target v10 deployment:

- [Installing the v10 target on Kubernetes.](#)
- [Installing the v10 target system on OpenShift and Cloud Pak for Integration.](#)
- [Installing the v10 target system on VMware.](#)

---

## Extracting the v2018 source data on VMware

Run the `save_v2018_source_configuration.py` script to gather the source deployment's configuration settings.

## Before you begin

---

Complete the steps described in [Migration preparation](#).

## About this task

---

The operations in the task are run on the Management and Portal VMs where you configured the migration scripts and tools, as described in [Migration preparation](#). Note: It is recommended that you keep the console output of all python scripts that are run. On most MacOS and Linux environments, appending the following to the end of the command saves the output to the specified file:

```
2>&1 | tee <output filename>
```

## Procedure

---

Management subsystem

1. SSH to your VM as the `apicadm` user, run `sudo -i`, change to your project directory that you created in [Migration preparation](#), and add the directory to your PATH variable.

```
ssh apicadm@<VM hostname>
sudo -i
cd /root/ffm/<project dir>
export PATH=/root/ffm/<project dir>:$PATH
```

2. Load the extract image file into the VMs local docker registry. Run:

- On v2018.4.1.20 and previous releases:

- `# docker load -i /root/ffm/<project dir>/build/v2018-dr-upgrade/extract-image.tar.gz`

- Verify that the image is loaded with `docker images`

```
docker images
REPOSITORY TAG
IMAGE ID CREATED SIZE
apiconnect/ibm-apiconnect-management-v2018-to-v10-extract v10.0.5-21-EXTRACT
24d49cf227cb 5 weeks ago 939MB
...
```

- Take note of the version in the TAG column.

- On v2018.4.1.24 and later releases:

- Uncompress the image file:

```
gzip -d extract-image.tar.gz
```

- Load the image file:

```
ctr --namespace=k8s.io images import extract-image.tar
```

- List the images to confirm successful load:

```
ctr -n k8s.io images ls |grep v2018-to-v10-extract
```

```
docker.io/apiconnect/ibm-apiconnect-management-v2018-to-v10-extract:v10.0.5-38-EXTRACT
```

- Take note of the image version in output.

3. Verify and fix any inconsistencies in the Management database by running the `apicops preupgrade` command:

```
apicops preupgrade -m fix --upgrade_to v10 -l apicopsOutput.txt
```

This commands checks for known data issues and cleans up the data. The time the command takes to run depends on the size of the management database. `apicops` reports any problems that it is unable to fix at the end of the `apicopsOutput.txt` output file. If `apicops` reports any problems, open a support case and include the `apicopsOutput.txt` file, do not attempt the migration until the case is resolved.

4. Run the extraction script:

```
python3 save_v2018_source_configuration.py -n default -u admin -p <password> -r admin/default-idp-1 -s <management hostname> -storage_class local-storage -extract_image <image path>:<version>-EXTRACT -skip_ptl -skip_a7s -skip_gw
```



Where

- `<password>` is the Cloud Manager UI admin password.
- `<management VM>` is the fully qualified hostname of your Management VM (the endpoint for your Management admin API).
- `<image path>` is the path to the image file:
  - On v2018.4.1.20 and earlier releases, `<image path>` is:

```
apiconnect/ibm-apiconnect-management-v2018-to-v10-extract
```

- On v2018.4.1.24 and later releases, `<image path>` is:

```
docker.io/apiconnect/ibm-apiconnect-management-v2018-to-v10-extract
```

- `<version>` is the version of the extract image, as noted in step 2.

Note: The extract script must be run within 2 hours of running the `apicops` health check.

5. The save script creates a `data` directory and two yaml files `gateway_portal_mapping.yaml` and `provider_org_credentials.yaml`. Copy the `data` directory and the two yaml files to your v10 `apicv10install` directory.

```
tar -cvf data_mgmt.tar data
scp data_mgmt.tar <username>@<servername>:<path>
scp gateway_portal_mapping.yaml <username>@<servername>:<path>
scp provider_org_credentials.yaml <username>@<servername>:<path>
```

Portal subsystem

6. SSH to your Portal VM as the `apicadm` user, run `'sudo -i'`, change to your project directory, and add the directory to your `PATH` variable.

```
ssh apicadm@<VM hostname>
sudo -i
cd /root/ffm/<project dir>
export PATH=/root/ffm/<project dir>:$PATH
```

7. From the project directory, run:

```
python3 save_v2018_source_configuration.py -ptl_ns default -skip_mgmt -skip_gw -skip_a7s
```

The script creates a `data` directory where all the extracted data is stored.

8. Upon completion of the script, copy the new `data` directory off the Portal VM to your v10 `apicv10install` directory.

```
tar -cvf data_portal.tar data
scp data_portal.tar <username>@<servername>:<path>
```

9. In your `apicv10install` directory, merge the two `data` directories that you copied from the Portal and Management VMs. The contents of this `data` directory should appear as shown:

```
$ tar -xf data_portal.tar
$ tar -xf data_mgmt.tar
$ ls -l data
total 20
-rw-r----- 1 <user> <user> 541 Oct 24 15:32 config_portal.yaml
-rw-r----- 1 <user> <user> 2552 Oct 24 17:19 config.yaml
drwxr-x--- 3 <user> <user> 4096 Oct 24 17:19 <management deployment name>
drwxr-x--- 2 <user> <user> 4096 Oct 24 15:29 <portal deployment name>
drwxr-x--- 2 <user> <user> 4096 Oct 24 17:19 temp
```

you should also have the two yaml files from Management subsystem extraction in the same location as the `data` directory:

```
$ ls -l
total 116452
drwxr-x--- 5 <user> <user> 4096 Oct 24 17:19 data
-rw-r----- 1 <user> <user> 2505 Oct 25 09:49 gateway_portal_mapping.yaml
-rw-r----- 1 <user> <user> 1146 Oct 25 09:49 provider_org_credentials.yaml
```

## What to do next

---

Install your target v10 deployment:

- [Installing the v10 target on Kubernetes.](#)
- [Installing the v10 target system on OpenShift and Cloud Pak for Integration.](#)
- [Installing the v10 target system on VMware.](#)

## Installing the v10 target on Kubernetes

---

Prepare your Kubernetes environment, create your API Connect secrets, install v10.

## About this task

---

Follow this task along with the Kubernetes installation instructions: [Deploying on Kubernetes](#). The installation of v10 for v2018 form factor migration differs from a standard v10 installation as follows:

- You must use the v2018 Management encryption secret from your v2018 source deployment (a new key will be generated for the Portal subsystem).
- You must not use the same Management subsystem backup location and path as you use for your v2018 source deployment.
- You can use the same Portal subsystem backup location and path as you use for your v2018 source deployment. Reuse the v2018 portal backup path if you want to restore your v2018 portal site backups - to migrate your site customizations.

- Ensure that any network route between the target installation and the source system is disabled. The target system must not be able to communicate with the source system.

This task describes the steps of the standard v10 installation that must be followed, and at which points in the process to apply the v2018 extracted data.

Note: It is recommended that you keep the console output of all python scripts that are run. On most MacOS and Linux environments, appending the following to the end of the command saves the output to the specified file:

```
2>&1 | tee <output filename>
```

## Procedure

1. Obtain the API Connect v10 product files: [Obtaining product files](#).
2. Create your v10 namespaces, deploy the API Connect v10 operators, and cert-manager. The steps differ depending on whether you use one namespace for all API Connect subsystems, or whether you want different namespaces for each subsystem. In both cases, complete all steps on these pages except for the last step of installing your API Connect subsystems:
  - [Deploying operators in a single-namespace API Connect cluster](#).
  - [Deploying operators in a multi-namespace API Connect cluster](#).

Important: Return to this page and complete [#ffm\\_installing\\_2018-to-10\\_k8s\\_create\\_sec](#) and [4](#) before installing the API Connect subsystems.

3. Create the secrets based on your v2018 extracted data.

- a. Create the secrets by running the `create_secrets_in_target.py` script:

- If both Management and Portal are in the same namespace, or if Portal is not installed:

```
python3 create_secrets_in_target.py -n <namespace>
```

- If Management and Portal are in different namespaces:

```
python3 create_secrets_in_target.py -mgmt_ns <mgmt_namespace> -ptl_ns <portal_namespace>
```

When the script detects more than one portal subsystem, it prompts you to select the portal subsystem name that you want to create the secrets for. For the remaining portal subsystems, specify the different portal namespace and name with the `-ptl_ns` and `-ptl_subsys_name` arguments:

```
python3 create_secrets_in_target.py -skip_mgmt -ptl_ns <portal_x_namespace> -ptl_subsys_name <portal subsystem name>
```

- b. Verify that the secrets were created.

Run the `kubectl get secrets` command as in the following example:

```
kubectl get secrets -n <namespace>
NAME TYPE DATA AGE
...
<mgmt deployment name>-enc-key Opaque 1 2m47s
<portal deployment name>-enc-key Opaque 1 2m47s
management-backup-secret Opaque 2 2m47s
portal-backup-secret Opaque 2 2m47s
```

- c. Delete the v2018 Portal database encryption secret `<ptl_deployment_name>-enc-key`.

Run the following command to delete the secret:

```
kubectl -n <namespace> delete secret <portal deployment name>-enc-key
```

Note: The v2018 Portal database encryption secret is not compatible with v10. After you delete the `<ptl_deployment_name>-enc-key`, a new secret will be generated when you apply the subsystem CR in a later step.

4. Edit the subsystem YAML files as described in [Installing the API Connect subsystems](#). You must update the following properties with the data that is extracted from your source v2018 deployment, which can be found in `data/config.yaml` for the Management subsystem, and `data/config_portal.yaml` for the Portal. Important: If you do not use the configuration from the source system, the migration fails.

- Management subsystem properties:

```
encryptionSecret:
 secretName: mgmt-enc-key
```

- Portal database backup details,:

```
portalBackup:
 credentials: portal-backup-secret
 host: <remote backup server IP>
 path: <example path>
 port: <remote backup server path>
 protocol: sftp
```

5. Continue with the installation of the subsystems, as described in [Installing the API Connect subsystems](#).

Note: To migrate all your v2018 data, install at least as many Portal and Gateway subsystems as you have on your v2018 source deployment. If you want to do a staged migration, you can install any remaining subsystems later.

Postinstallation verification and v10 toolkit CLI download.

6. Log in to the Cloud Manager UI: [Login to Cloud Manager UI](#).
7. Download the v10 toolkit CLI (CLI only) and `credentials.json` file, see [Installing the v10 toolkit](#).
8. Unpack the v10 toolkit CLI, and copy the `apic-slim` and `credentials.json` files to your `apicv10install` directory. Rename `apic-slim` to `apic`.
9. Ensure that your v10 toolkit CLI `apic` command is in your PATH variable:

```
EXPORT PATH=<apicv10install directory>:$PATH
```

10. Set your toolkit CLI credentials with:

```
apic client-creds:set <apicv10install directory>/credentials.json
```

11. Verify you can log in to your Management subsystem with the toolkit CLI. Use the same password as you used to log in to the Cloud Manager UI.

```
apic login --server <platform_api_endpoint> --username admin --realm admin/default-idp-1
```

where `<platform_api_endpoint>` is the FQDN of the platform api endpoint that is defined for your v10 Management subsystem, as returned by this command:

```
kubectl get ingress -n <namespace> | grep platform-api
```

For more information, see [v10 toolkit CLI login](#)

## What to do next

Restore your v2018 source data to your v10 target deployment: [Restoring the v2018 data to the v10 target deployment](#).

# Installing the v10 target system on OpenShift and Cloud Pak for Integration

On the target system, configure a cluster, create the necessary secrets, and install API Connect.

## Before you begin

Check that your `apicv10install` directory contains:

- The migration scripts, as described in [v10 target deployment preparation](#).
- The `data` directory that contains the v2018 extracted data.

## About this task

The installation of v10 for v2018 form factor migration differs from a standard installation in that you must use the v2018 encryption secret for your Management subsystem, instead of the default encryption secret that is created in a standard v10 installation. You can also use the same portal backup configuration if you want to restore your v2018 portal backup to v10 to keep your v2018 portal site customizations. This task describes the steps of the standard v10 installation that should be followed, and at which points in the process to apply the v2018 extracted data.

All operations in this task are completed from your `apicv10install` directory.

Note: It is recommended that you keep the console output of all python scripts that are run. On most MacOS and Linux environments, appending the following to the end of the command saves the output to the specified file:

```
2>&1 | tee <output filename>
```

## Procedure

1. Install the API Connect operators and create your namespace. Follow the steps described on [Installing operators](#).
2. Create the secrets based on your v2018 extracted data.

a. Create the secrets by running the `create_secrets_in_target.py` script:

- If both Management and Portal are in the same namespace, or if Portal is not installed:

```
python3 create_secrets_in_target.py -n <namespace>
```

- If Management and Portal are in different namespaces:

```
python3 create_secrets_in_target.py -mgmt_ns <mgmt_namespace> -ptl_ns <portal_namespace>
```

When the script detects more than one portal subsystem, it prompts you to select the portal subsystem name that you want to create the secrets for. For the remaining portal subsystems, specify the different portal namespace and name with the `-ptl_ns` and `-ptl_subsys_name` arguments:

```
python3 create_secrets_in_target.py -skip_mgmt -ptl_ns <portal_x_namespace> -ptl_subsys_name <portal subsystem name>
```

b. Verify that the secrets were created.

Run the `oc get secrets` command as in the following example:

```
oc get secrets -n <namespace>
NAME TYPE DATA AGE
...
<mgmt deployment name>-enc-key Opaque 1 2m47s
<portal deployment name>-enc-key Opaque 1 2m47s
management-backup-secret Opaque 2 2m47s
portal-backup-secret Opaque 2 2m47s
```

c. Delete the v2018 Portal database encryption secret `<ptl_deployment_name>-enc-key`.

Run the following command to delete the secret:

```
oc -n <namespace> delete secret <portal deployment name>-enc-key
```

Note: The v2018 Portal database encryption secret is not compatible with v10. After you delete the `<ptl_deployment_name>-enc-key`, a new secret will be generated when you apply the subsystem CR in a later step.

3. Cloud Pak for Integration: Install the Platform UI, as described in <https://www.ibm.com/docs/en/cloud-paks/cp-integration>.

4. Create a top-level CR YAM: file, but do not apply it: run `install_apic_on_ocp.py` with `-no_install`:

On OpenShift, run:

```
python3 install_apic_on_ocp.py -name <name> -n <apic-namespace> -no_install -license "<license>" --storageclass_apic
<storage class>
...
Create top level CR::
no_install flag is set to true. Not installing API Connect.
Location of API Connect top level CR file : /root/installapic/data/top_level_cr.yaml
...
```

This script performed the following actions in the target cluster:

\*\*\*\*\*

- Generated the yamls for catalog sources, subscriptions and top level CR. Check the data directory and data/temp directory.

...

Where

- `<name>` is the name that you want for your v10 top-level CR deployment. If not specified the default is `minimum`.
- `<namespace>` is the namespace that you created for your v10 API Connect.
- `<license>` is the license ID, see [API Connect licenses](#).
- `<storage class>` is the name of the storage class your v10 API Connect deployment is using.

On Cloud Pak for Integration you must specify your Platform UI storage class and license: `-cp4i -storage_class_pn`

`<platform ui storage class> -license_pn <platform ui license>`

```
python3 install_apic_on_ocp.py -name <name> -n <namespace> -no_install -license "<license>" -storageclass_apic <storage class> -cp4i -storage_class_pn <platform ui storage class> -license_pn <platform ui license>
```

The `top_level_cr.yaml` is created in your data directory.

Important: Save the generated Portal encryption key

5. Edit the generated `data/top_level_cr.yaml` file. You must set the `management.databaseBackup` section to your new location for v10 backups. You can change other settings in this file, but do not change the `management.encryptionSecret`. Example `top_level_cr.yaml` file:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APICConnectCluster
metadata:
 labels:
 app.kubernetes.io/instance: apiconnect
 app.kubernetes.io/managed-by: ibm-apiconnect
 app.kubernetes.io/name: apiconnect-<name>
 name: <name>
 namespace: <namespace>
spec:
 analytics:
 storage:
 enabled: true
 type: unique
 license:
 accept: true
 license: <license>
 use: nonproduction
 management:
 customApplicationCredentials: []
 databaseBackup:
 credentials: management-backup-secret
 host: backup.ibm.com
 path: /backups/management
 port: 22
 protocol: sftp
 schedule: 0 0 1 1 *
 encryptionSecret:
 secretName: v2018source-management-enc-key
 portal:
 portalBackup:
 credentials: portal-backup-secret
 host: backup.ibm.com
 path: /backups/portal
 port: 22
 protocol: sftp
 schedule: 0 0 1 1 *
 profile: nlxc7.m48
 storageClassName: <storage class>
 version: 10.0.x.x
```

6. Apply the `top_level_cr.yaml` file:

```
oc apply -f top_level_cr.yaml
```

Your v10 API Connect cluster is deployed. You can monitor the progress with:

```
oc get all -n <apic-namespace>
```

7. Install additional Portal, Gateway, and Analytics subsystems as needed, as individual subsystem CRs.

Note: To migrate all your v2018 data, install at least as many Portal and Gateway subsystems as you have on your v2018 source deployment. If you want to do a staged migration, you can install any remaining subsystems later.

Postinstallation verification and v10 toolkit CLI download.

8. Log in to the Cloud Manager UI: [Login to Cloud Manager UI](#).
9. Download the v10 toolkit CLI (CLI only) and `credentials.json` file, see [Installing the v10 toolkit](#).
10. Unpack the v10 toolkit CLI, and copy the `apic-slim` and `credentials.json` files to your `apicv10install` directory. Rename `apic-slim` to `apic`.
11. Ensure that your v10 toolkit CLI `apic` command is in your PATH variable:

```
EXPORT PATH=<apicv10install directory>:$PATH
```

12. Set your toolkit CLI credentials with:

```
apic client-creds:set <apicv10install directory>/credentials.json
```

13. Verify you can log in to your Management subsystem with the toolkit CLI. Use the same password as you used to log in to the Cloud Manager UI.

```
apic login --server <platform_api_endpoint> --username admin --realm admin/default-idp-1
```

where `<platform_api_endpoint>` is the FQDN of the platform api endpoint that is defined for your v10 Management subsystem, as returned by this command:

```
oc get routes -n <apic namespace> | grep platform-api
```

For more information, see [v10 toolkit CLI login](#).

## What to do next

Restore your v2018 source data to your v10 target deployment: [Restoring the v2018 data to the v10 target deployment](#).

## Installing the v10 target system on VMware

Install your v10 OVA deployments and prepare them for restoring your v2018 data.

## About this task

Follow the standard VMware installation procedure, but set the encryption-secret for your v10 deployment to be the same as the encryption-secret extracted from your v2018 source deployment. Copy your `apicv10install` directory to your v10 VMs, and prepare the v2018 data load image.

## Procedure

1. Install your v10 API Connect target deployment. Follow the standard v10 installation steps: [Install API Connect on VMware](#), but with the following differences:
  - Ensure you specify different Management backup settings from your v2018 source deployment.
  - If you want to restore portal backups, set the portal backup settings the same as on your v2018 source.
  - **IMPORTANT:** Before you create the ISO files for your Management VM, set the encryption secret to the one you extracted from your v2018 source deployment:

```
apicup certs set <subsystem name> encryption-secret --encoded-string <v2018 enc secret>
```

Where:

- `<subsystem name>` is the name of your Management subsystem.
- `<v2018 enc secret>` is the v2018 encryption secret from your source Management deployment, located in `data/<mgmt_subsystem_name>/<mgmt_subsystem_name>-enc-key.yaml` file; for example:

```
apiVersion: v1
data:
 encryption_secret.bin: Xv2ES/0aIZnxwMdnGYsrfNM6c3S5iQ2qDbsfjL5bJlT7En0DdM7qlbKMWJTeSbAUVK2AXaMI1NCGV+tNX6ASn2
 kind: Secret
...
```

Important: If you do not set the encryption secret before you created the ISO files, you must create the ISO files again. You cannot update the encryption secret after you deploy the management VM.

Postinstallation verification and v10 toolkit CLI download.

2. Log in to the Cloud Manager UI: [Login to Cloud Manager UI](#).
3. Download the v10 toolkit CLI (CLI only) and `credentials.json` file, see [Installing the v10 toolkit](#).
4. Unpack the v10 toolkit CLI, and copy the `apic-slim` and `credentials.json` files to your `apicv10install` directory. Rename `apic-slim` to `apic`.
5. SSH to your Management VM and Portal VMs, copy over your `apicv10install` directory, and set it as the first entry in your PATH variable:

```
ssh apicadm@<VM hostname>
sudo -i
scp -r <apicv10install directory> /root
export PATH=/root/<apicv10install directory>:$PATH
```

Note: If you are deploying a three node cluster, you need to copy the `apicv10install` and run the load commands on just one node in each cluster.

6. Set your toolkit CLI credentials with:

```
apic client-creds:set <apicv10install directory>/credentials.json
```

7. Verify you can log in to your Management subsystem with the toolkit CLI. Use the same password as you used to log in to the Cloud Manager UI.

```
apic login --server <platform_api_endpoint> --username admin --realm admin/default-idp-1
```

where `<platform_api_endpoint>` is the FQDN of the platform api endpoint that is defined for your v10 Management subsystem, this can be found in your `apiconnect-up-v10.yml` file in your project directory. For more information, see [v10 toolkit CLI login](#).

8. On your Management VM, prepare the full path string of the migration load image `ibm-apiconnect-management-v2018-to-v10-load`.

```
cd /var/lib/apiconnect-subsystem/registry/<apic version>/ibm-apiconnect-management-v2018-to-v10-load/<version>-LOAD
```

Get the `sha256sum` of the `manifest.json` file. Example:

```
ls -l
...
-rw-r--r-- 1 apicadm apicadm 3254 Sep 16 21:36 manifest.json
-rw-r--r-- 1 apicadm apicadm 33 Sep 16 21:36 version

sha256sum manifest.json
e9360e36f2544c84e83d2b0be7e2b47f9dc223dee2f604ee8e98778a01302d32 manifest.json
```

Use the `sha256sum` to assemble the full image name:

```
127.0.0.1:8675/<apic version>/ibm-apiconnect-management-v2018-to-v10-load@sha256:<sha256sum of manifest.json>
```

## What to do next

Restore your v2018 source deployment data to your v10 target deployment: [Restoring the v2018 data to the v10 target deployment](#).

# Restoring the v2018 data to the v10 target deployment

Restore the saved configuration and backups from the source deployment onto the target deployment.

## Before you begin

- Complete the installation of v10 for your target platform:
  - [Installing the v10 target system on OpenShift and Cloud Pak for Integration](#).
  - [Installing the v10 target on Kubernetes](#).
  - [Installing the v10 target system on VMware](#)
- If you want to migrate any changes on your v2018 deployment that happened since you did the extract process, repeat the extract steps:
  - [Extracting the v2018 source data on VMware](#).
  - [Extracting the v2018 source data on Kubernetes and OpenShift](#).
- Ensure that any network route between the target installation and the source system is disabled. The target system must not be able to communicate with the source system.
- **OVA Users:** All operations are run from the `apicv10install` directory, which you copied to your VMs after installation: [Installing the v10 target system on VMware](#). SSH to your target VMs as the `apicadm` user, run `'sudo -i'`, and `cd` to this directory:

```
ssh apicadm@<VM hostname>
sudo -i
cd /root/<apicv10install directory>
```

- Ensure you have the v10 toolkit CLI setup in your `apicv10install` directory, see [Installing the v10 toolkit](#). Set your PATH variable as follows:

```
EXPORT PATH=<full path of apicv10install directory>:$PATH
```

Verify that you can log in to the Management server as the Cloud Manager admin user with the toolkit CLI:

```
apic login --server <platform_api_endpoint> --username admin --realm admin/default-idp-1
```

where `<platform_api_endpoint>` is the FQDN of the platform api endpoint that is defined for your v10 Management subsystem. To find this:

- OpenShift: Run

```
oc get routes -n <apic namespace> | grep platform-api
```

- Kubernetes: Run

```
kubectl get ingress -n <apic namespace> | grep platform-api
```

- OVA: The platform api endpoint is defined in your `apiconnect-up-v10.yml` file.

The password is the same as the admin user password for the Cloud Manager UI, and is set to the default for a new v10 installation on your platform. For more information, see [v10 toolkit CLI login](#).

- Update your `gateway_portal_mapping.yaml` mapping file in your `apicv10install` directory, adding your v10 endpoints. For any subsystems that you do not want to migrate, leave them set to the placeholder value. The example below shows a mapping file where the gateway service called `v5gway_service_1` is not migrated.

```
analytics_mapping:
 analytics_service1:
 https://analytics_ingestion.v2018.example.com: https://analytics_ingestion.v10.example.com
gateway_mapping:
 apigway_service1:
 https://gateway.v2018.example.com: https://gateway.v10.example.com
 https://gateway-manager.v2018.example.com: https://gateway-manager.v10.example.com
v5gway_service_1:
 https://gw.source-apic.fyre.example.com: https://NEW_GATEWAY_ENDPOINT_HOST_IN_TARGET_SYSTEM
 https://gwd.source-apic.fyre.example.com: https://NEW_GATEWAY_API_ENDPOINT_BASE_HOST_IN_TARGET_SYSTEM
portal_mapping:
 portal_service_1:
 https://portal-director.v2018.example.com: https://portal-director.v2018.example.com
 https://portal-web.v2018.example.com: https://portal-web.v2018.example.com
```

You can find your new v10 endpoints with:

- OpenShift:

```
oc get routes -n <apic namespace>
```

- Kubernetes:

```
kubectl get ingress -n <apic namespace>
```

- OVA: Endpoints are defined in your `apiconnect-up-v10.yml` file.

Note: On the gateway appliance, the Gateway Manager URL uses port `3000`. The port number is appended to the v10 endpoint in the mappings file. For example, `https://gateway-manager.v10.example.com:3000`.

Unmigrated subsystems can be migrated later, see [Migrating additional v2018 subsystems: Staged migration](#).

Note: If you want your v10 target deployment to use the same endpoint URLs as your v2018 source deployment, you can skip updating this file.

## About this task

In this task, migration scripts are used to:

- Restore the Management subsystem.
- Register Gateways and Portals on the target system.
- Update Portals in the catalog to point to new Portal systems.
- Restore Portal backups for each site.
- Update Gateways in the catalog to point to new Gateway systems.

All operations in this task are run from your `apicv10install` directory.

Note: It is recommended that you keep the console output of all python scripts that are run. On most MacOS and Linux environments, appending the following to the end of the command saves the output to the specified file:

```
2>&1 | tee <output filename>
```

## Procedure

1. Verify that all pods in your v10 target deployment are running and no errors are reported:

- Kubernetes:

```
kubectl get pods -n <namespace>
```

- OpenShift:

```
oc get pods -n <namespace>
```

- OVA:

```
kubectl get pods
```

2. **OpenShift and Cloud Pak for Integration only:** Run the following command:

```
oc adm policy add-scc-to-user anyuid system:serviceaccount:<apic namespace>:default
```

Where `<apic namespace>` is your v10 API Connect namespace.

3. Run the load script:

```
python3 load_v2018_data_to_v10.py -storage_class <storage_class> -registry_secret <image_pull_secret> -load_storage_size 50 -load_image <load_image_url> -n <apic_namespace> -operator_namespace <apic operator namespace> -s <platform_api_endpoint> -silent
```

Where

- `<storage_class>` is the name of the storage class your v10 API Connect deployment is using. **OVA users:** This must be set to `local-storage`.
- **Non-OVA users:** `<image_pull_secret>` is the secret that is used to pull the API Connect images. Not needed for OVA. For OpenShift, the cluster default is used if not specified.
- `<load_image_url>` is the full name and path of the `ibm-apiconnect-management-v2018-to-v10-load` image. On OVA this URL is obtained as described in the installation steps: [Get OVA load image URL](#)
- `<apic_namespace>` is the namespace where your v10 API Connect subsystem instances are installed. **OVA users:** This must be set to `default`.
- `<apic operator namespace>` is the namespace where you v10 API Connect operators are installed. **OVA users:** This must be set to `default`.
- `<platform_api_endpoint>` is the FQDN of the platform api endpoint that is defined for your v10 Management subsystem. To find this:
  - OpenShift: Run

```
oc get routes -n <apic namespace> | grep platform-api
```
  - Kubernetes: Run

```
kubectl get ingress -n <apic namespace> | grep platform-api
```
  - OVA: The platform api endpoint is defined in your `apiconnect-up-v10.yml` file.

The script takes some time to complete. Check the status of the pods in your API Connect subsystem namespace and check for any pods in Error state. The script attempts to pull the API Connect migration load image from your image repository, failure to pull the image can be observed by checking the pods:

- Kubernetes:

```
kubectl get pods -n <namespace>
```

- OpenShift:

```
kubectl get pods -n <namespace>
```

- OVA:

```
kubectl get pods
```

If see output such as:

```
v2018-mgmt-load-bc2df 0/1 ImagePullBackOff 0 2m4s
```

terminate the load script and investigate why the image cannot be loaded. When the load script is rerun it cleans up failed pods from previous attempts. OVA users: if the problem is not clear, for example an incorrect load image path, gather logs and open a support case.

Note: The v2018 load script overwrites any data that you configured on your v10 target deployment, with the v2018 source data. For example, the admin password for your v10 Cloud Manager UI is replaced with the one from your v2018 source deployment.

4. If you want your v10 target deployment to use the same endpoint URLs as your v2018 source deployment, skip to step [3](#).
5. Register Gateways and Portals on the target system.

Run the `register_gateway_portals_in_target.py` script:

OVA users: On your Management VM, run:

```
python3 register_gateway_portals_in_target.py -s <platform_api_endpoint> -u admin -p <password> -r admin/default-idp-1 -n default -silent
```

Where

- <platform\_api\_endpoint> is the FQDN of the platform api endpoint that is defined for your v10 Management subsystem.
- <password> is the Cloud Manager UI admin user password, from your v2018 deployment.

Kubernetes, OpenShift, and Cloud Pak for Integration users:

- If Management, Portal and Gateway subsystems are in the same namespace:

```
python3 register_gateway_portals_in_target.py -s <platform_api_endpoint> -u admin -p <password> -r admin/default-idp-1 -n <apic_namespace> -silent
```

Where

- <platform\_api\_endpoint> is the FQDN of the platform api endpoint that is defined for your v10 Management subsystem.
  - <password> is the Cloud Manager UI admin user password, from your v2018 deployment.
  - <apic\_namespace> is the namespace where your v10 API Connect subsystem instances are installed.
- If Management, Portal and Gateway subsystems are in different namespaces, use -mgmt\_ns, -ptl\_ns, and -gw\_ns to specify each, for example:

```
python3 register_gateway_portals_in_target.py -s <platform_api_endpoint> -u admin -p <password> -r admin/default-idp-1 -mgmt_ns <mgmt_namespace> -ptl_ns <portal_namespace> -gw_ns <gateway_namespace> -silent
```

- If you are using SSO authentication, replace:

```
-u <username> -p <password>
```

with:

```
--sso
```

- Multiple namespaces can alternatively be specified with "ns1|ns2|ns3" notation. For example:

```
python3 register_gateway_portals_in_target.py -n "ns1|ns2|ns3" -u admin -p <password> -s <platform_api_endpoint> -r admin/default-idp-1 -silent
```

```
python3 register_gateway_portals_in_target.py -s PLATFORM_API_HOSTNAME -u admin -p <password> -r admin/default-idp-1 -mgmt_ns ns1 -ptl_ns "ns1|ns2" -gw_ns "ns1|ns2|ns3" -silent
```

Note: If you run `register_gateway_portals_in_target.py` for a Cloud Pak for Integration target system, and you see the error

**The API Connect Gateway Service is already registered to another API Manager instance**

restart all the gateway pods, wait for them to reach **Ready** state, and run `register_gateway_portals_in_target.py` again.

6. Run `update_to_new_portals.py` to update portals defined in your catalogs to point to your new Portal subsystems.

Note: If your source system has no portal services, skip this step. Also, skip this step if no portal sites are created for any of your catalogs.

OVA users: On your Management VM (not your Portal VM), run:

```
python3 update_to_new_portals.py -s <platform_api_endpoint> -u admin -p <password> -r admin/default-idp-1 -n default -api_manager_hostname <platform_api_endpoint> -silent
```

Kubernetes, OpenShift, and Cloud Pak for Integration users:

- If the Management and Portal subsystems are in the same namespace:

```
python3 update_to_new_portals.py -s <platform_api_endpoint> -u admin -p <password> -r admin/default-idp-1 -n <apic_namespace> -api_manager_hostname <platform_api_endpoint> -silent
```

- If the Management and Portal subsystems are in different namespaces:

```
python3 update_to_new_portals.py -s <platform_api_endpoint> -u admin -p <password> -r admin/default-idp-1 -mgmt_ns <mgmt_namespace> -ptl_ns <portal_namespace> -gw_ns <gateway_namespace> -silent
```

- If you are using SSO authentication, replace:

```
-u USERNAME -p PASSWORD
```

with:

```
--sso
```

7. Run `update_to_new_gateways.py` to update Gateways in the catalog to point to new Gateway systems.

Note: The response might indicate that you should run a health check but that is not needed because you will run a health check later as part of the verification task.

OVA users: On your Management VM, run:

```
python3 update_to_new_gateways.py -s <platform_api_endpoint> -u admin -p <password> -r admin/default-idp-1 -n default -api_manager_hostname <platform_api_endpoint> -silent
```

Kubernetes, OpenShift, and Cloud Pak for Integration users:

- If the Management and Gateway subsystems are in the same namespace:

```
python3 update_to_new_gateways.py -s <platform_api_endpoint> -u admin -p <password> -r admin/default-idp-1 -n <apic_namespace> -api_manager_hostname <platform_api_endpoint> -silent
```

- If the Management and Gateway subsystems are in different namespaces:

```
python3 update_to_new_gateway.py -s <platform_api_endpoint> -u admin -p <password> -r admin/default-idp-1 -mgmt_ns <mgmt_namespace> -gw_ns <gateway_namespace> -silent
```

- If you are using SSO authentication, replace:

```
-u USERNAME -p PASSWORD
```

with:



--sso

#### 8. Restore Portal backups for each site.

Note: If your source system has no portal services, skip this step. Also, skip this step if no portal sites are created for any of your catalogs.

To restore the Portal sites on the target system, you can either use the script `restore_portal_db.py` to restore all sites, or restore sites individually with the `restore_site` command.

Restoring all sites with the restore script:

- OVA users: On your Portal VM, run:

```
python3 restore_portal_db.py -n default
```

- Non-OVA users: Run the following:

```
python3 restore_portal_db.py -n <portal_subsystem_namespace>
```

Note: If there are more than one Portal subsystems on the target system, run the script for each of the matching source and target Portal subsystems with the `-ptl_ns` and `-ptl_subsys_name` arguments:

```
python3 restore_portal_db.py -ptl_ns <portal_namespace> -ptl_subsys_name <portal_subsystem_name>
```

Restoring sites individually, using `restore_site`. For OVA users, these steps should be run from the Portal VM:

##### a. Identify the portal `www` pod:

- OpenShift:

```
oc get pods -n <portal_subsystem_namespace> | grep www
```

- Kubernetes:

```
kubectl get pods -n <portal_subsystem_namespace> | grep www
```

- OVA:

```
oc get pods | grep www
```

##### b. Exec into the portal `www` pod's `admin` container:

- OpenShift:

```
oc exec -it -n <portal_subsystem_namespace> <www_pod> -c admin -- bash
```

- Kubernetes:

```
kubectl exec -it -n <portal_subsystem_namespace> <www_pod> -c admin -- bash
```

- OVA:

```
kubectl exec -it <www_pod> -c admin -- bash
```

##### c. List the site backups:

```
remote_transfer_backup -l
```

##### d. Download each site backup, and run it for all sites:

```
remote_transfer_backup -d <site_backup_filename>
```

##### e. Restore each of the site backups:

```
restore_site -f -u <site_url> <site_backup_filename>
```

Post-restore verification

#### 9. Check that your v10 migration was successful and remove any v2018 subsystems or data that you do not want: [Verifying the v10 target deployment](#).

Note: If you are doing a staged migration, it is recommended you do the verification before and after you add further subsystems: [Migrating additional v2018 subsystems: Staged migration](#).

---

## Verifying the v10 target deployment

Verify that the target deployment is working correctly. Remove any unwanted v2018 data and subsystems.

## Before you begin

Complete [Restoring the v2018 data to the v10 target deployment](#).

## Procedure

#### 1. From your `apicv10install` directory, use the `health_check_post_migration.py` script to health check the target system. See the following examples:

- If all the subsystems are in the same namespace, and the deployment uses username and password authentication (this applies to OVA):

```
python3 health_check_post_migration.py -s <platform_api_endpoint> -r admin/default-idp-1 -u admin -p <password> -n <apic_namespace>
```

Where:

- `<platform_api_endpoint>` is the FQDN of the platform api endpoint that is defined for your v10 Management subsystem.
- `<password>` is the Cloud Manager UI admin user password that you used on your v2018 deployment.
- `<apic_namespace>` is the namespace where your v10 API Connect subsystem instances are installed. **OVA users:** This must be set to `default`.

To check the health of your portal sites, add `-check_portal_site_health` to the above command. Portal sites take time to restore, so if you recently completed the restore steps, the script `health_check_post_migration.py` run with the `-check_portal_site_health` argument might report that sites are not healthy.

- **Non-OVA users:** When all subsystems are in different namespaces, and the deployment uses username/password authentication:

```
python3 health_check_post_migration.py -s <platform_api_endpoint> -r admin/default-idp-1 -u admin -p <password> -mgmt_ns <mgmt_namespace> -gw_ns <gateway_namespace> -ptl_s <portal_namespace> -a7s_ns <analytics_namespace>
```

- **Non-OVA users:** When all subsystems are in the same namespace, and the deployment uses SSO authentication:

```
python3 health_check_post_migration.py -n <apic_namespace> --sso -s <platform_api_endpoint>
```

- **Non-OVA users:** When all subsystems are in the same namespace, and the deployment uses SSO authentication silent mode:

```
python3 health_check_post_migration.py -n <apic_namespace> --sso -s <platform_api_endpoint> -api_key <api_key_value> -silent
```

Where:

- `<api_key_value>` is the API-key that is used in Cloud Manager SSO authentication.

- **Non-OVA users:** When you have multiple subsystems across namespaces:

```
python3 health_check_post_migration.py -s <platform_api_endpoint> -r admin/default-idp-1 -u admin -p <password> -n "<namespace1>|<namespace2>|<namespace3>"
```

2. Complete any further configuration steps needed for your deployment:

a. If necessary, update redirect URLs in the authorization server.

If OIDC or SSO authentication is being used, the redirect URLs must be updated in the authorization server after migration. For example, for Google OIDC, you must add the new redirect URLs in Google Developer Console. The new redirect URLs can be found in the user registry in Cloud Manager.

b. Complete configuration for Analytics.

- If Analytics services are present on the target system, configure analytics backup settings.
- Use the Cloud Manager UI to associate the Analytics services with the Gateways.

c. If the target system is Cloud Pak for Integration, your Platform UI password is unchanged when you installed Platform UI, it is not updated with any v2018 migrated data.

d. Review any v2018 scripts or utilities that you have that use the toolkit CLI. If you want to use these scripts in v10, check if the scripts referred to your v2018 portal and gateway services by name. If it did use the service names and not the IDs, then update these scripts to use the new v10 names for these services.

3. Validate configuration data:

a. Validate the data in the target API Connect system by logging in to Cloud Manager and API Manager UIs.

b. Validate the APIs published in the Gateways.

c. Validate the Portal site URLs

d. If necessary, update the DNS-mapping information.

When all of the APIs are verified, you might need to update the DNS-mapping information, to redirect the calls to the new Gateways and Portals.

e. Delete any v2018 provider organization data (APIs, Products, Consumer organizations) that you do not want from the API Manager UI.

f. Delete any v2018 portal or gateway subsystems that you do not want from the Cloud Manager UI.

The v10 migration is now complete.

4. If you migrated all your subsystems, you can uninstall v2018 API Connect on the source system. If you are doing a staged migration, do not uninstall your v2018 source system until all your subsystems are migrated, see [Migrating additional v2018 subsystems: Staged migration](#).

---

## Migrating additional v2018 subsystems: Staged migration

If you did not migrate all your v2018 portal and gateway subsystems initially, and now want to migrate the remaining ones.

For adding gateways after initial migration, migrate your v2018 source data to them by following these steps:

1. Add the new Gateway and Analytics related endpoints to the `gateway_portal_mapping.yaml` file.
2. Register Gateways and Analytics on the target system. Complete Step 5 in [Restoring the v2018 data to the v10 target deployment](#)
3. Update gateways in the catalog to point to new Gateway systems. Complete Step 7 in [Restoring the v2018 data to the v10 target deployment](#)
4. Associate your new Analytics subsystems with your gateways in the Cloud Manager UI.

For adding portals after initial migration, migrate your v2018 source data to them by following these steps:

1. Add the new Portal related endpoints to the `gateway_portal_mapping.yaml` file.
2. Register portals on the target system. Complete Step 5 in [Restoring the v2018 data to the v10 target deployment](#)
3. Update portals in the catalog to point to new portal systems. Complete Step 6 in [Restoring the v2018 data to the v10 target deployment](#)
4. Restore portal backups for each site. Complete Step 8 in [Restoring the v2018 data to the v10 target deployment](#)

Repeat the health check and verification steps after migrating each subsystem, see [Verifying the v10 target deployment](#).

---

## Troubleshooting the form-factor migration from v2018

Troubleshoot problems with migrating from v2018 to v10 on a different form factor.

Topics:

- [Script help](#)
- [Restarting migration](#)
- [Factory reset of portal subsystem](#)
- [Avoid error when a gateway is already registered to another API manager](#)

## Script help

---

You can access help for all the migration scripts by using the `--help` argument:

```
python3 <script_file_name> --help
```

## Restarting migration

---

On the target system, if the `register_gateway_portals_in_target.py`, `update_to_new_portals.py`, or `update_to_new_gateways.py` scripts fail, run the load script [3](#) again, with the `-reset_gateway_portal` flag as shown:

```
python3 load_v2018_data_to_v10.py ... -reset_gateway_portal
```

This resets the portals and deletes the gateway pods.

Note: If you deleted the data directory after you registered the Gateways or Portals, but want to use the same target API Connect system that you already installed, then extract the data again:

- [Extracting the v2018 source data on VMware.](#)
- [Extracting the v2018 source data on Kubernetes and OpenShift.](#)

## Factory reset of portal subsystem

---

The migration load script with `-reset_gateway_portal` does a factory reset of your portal subsystems. Portals can also be factory reset by following these steps:

1. Log in to the portal `www` pod `admin` container:

```
kubectl exec -it <portal www pod> -c admin -n <apic namespace> -- bash
```

2. Run the command:

```
curl -X DELETE -k --cert /etc/nginx/ssl/portal-director-client-both.pem https://localhost:3009/v2/portal/factory-reset
```

3. Use the `list_sites` command to check that all sites are deleted.
4. Check health of the Portal subsystem.

## Avoid error when a gateway is already registered to another API manager

---

Error messages

```
Error: An error occurred communicating with the gateways subsystem at 'some_url' (status: 400, response: '"Unable to perform initial registration for API Management. Error: The API Connect Gateway Service is already registered to another API Manager instance"')
```

To fix this first try restarting the gateways, or if you are using Gateway appliances, restart them. If the problem persists, reload the v2018 data with the `load_v2018_data_to_v10.py` script and repeat the remaining steps of the migration process, starting with Step [5](#) in [Restoring the v2018 data to the v10 target deployment](#).

## Replacing passwords, keys, and certificates

---

Ensure the security of your API Connect deployment by replacing compromised passwords, keys, and certificates.

### About this task

---

Each subsystem uses one or more passwords, database keys, and registration secrets. In some cases, an update takes effect immediately or on the next log-in, but some components require a restart to apply the update. Follow the procedure for updating each subsystem to ensure that you complete all required steps.

All subsystems use TLS certificates to encrypt communications both within each subsystem, between subsystems, and with clients. The procedure for renewing a certificate depends on whether you use cert-manager. After you replace a certificate in a subsystem, you must restart all pods that are affected by the certificate, so there will be a service interruption.

The instructions in this section are written for deployments on Kubernetes and details for other platforms appear at the end of the section. Before you begin updating passwords, keys, and certificates, review the entire section to ensure you understand the procedures and are aware of any platform-specific differences that affect your deployment.

Attention: Before you change any passwords, keys, or certificates, you should disable scheduled backups for the Management subsystem as explained in this section. After the updates are complete, enable backups.

- [Disabling backups before updating secrets and certificates](#)  
Disable scheduled backups to ensure that you don't update secrets and certificates while a backup is running in your API Connect deployment.
- [Management subsystem: updating the password, keys, and secrets](#)  
Update the cloud administrator password, the key for encrypted database fields, and the `client_ID` and `client_secret` values for the Management subsystem in an API Connect deployment.
- [Gateway subsystem: updating the password and secret](#)  
Update the administrator password and the `client_secret` values for the Gateway subsystem in an API Connect deployment.
- [Portal subsystem: updating keys and secrets](#)  
Update the database passwords, keys, and the `client_secret` values for the Portal subsystem in an API Connect deployment.
- [Analytics subsystem: replacing the TLS profile and keys](#)  
Update the Analytics subsystem's TLS profile and keys for backup and restore operations in your API Connect deployment.

- [Renewing TLS certificates](#)  
Renew TLS certificates in your API Connect deployment.
- [Management subsystem: restarting pods](#)  
Restart Management subsystem pods in your API Connect deployment.
- [Gateway subsystem: restarting pods](#)  
Restart Gateway subsystem pods and components in your API Connect deployment.
- [Portal subsystem: restarting pods](#)  
Restart Portal subsystem pods in your API Connect deployment.
- [Analytics subsystem: restarting endpoints and pods](#)  
Restart the pods that support the Analytics service and endpoints in your API Connect deployment.
- [Re-enabling scheduled backups](#)  
If you disabled scheduled backups before updating keys, secrets, or passwords, enable the backups again for your API Connect deployment.

---

## Disabling backups before updating secrets and certificates

Disable scheduled backups to ensure that you don't update secrets and certificates while a backup is running in your API Connect deployment.

---

### About this task

Manually disable any configured backup schedule prior to updating and passwords, keys, profiles, secrets, or certificates. If a backup is taken while an update is running, then that backup should be considered potentially corrupted.

---

### Procedure

1. Disable backups by completing the following steps:
  - a. Open the management cluster CR for editing by running the following command:

```
kubectl -n namespace edit CR-name
```

where:

- **namespace** is the namespace where the Management subsystem is installed.
- **CR-name** is the name of the CR that was used to install the Management subsystem.

- b. In the **databaseBackup** section of the CR, remove the **schedule** entry.
  - c. Save and close.
2. Verify that no backups are running by completing the following steps:
    - a. List all your Management backups by running the following command:

```
kubectl -n namespace get managementbackup
```

where **namespace** is the namespace where the Management subsystem is installed.

- b. In the results, check the STATUS column to ensure that all backups are listed as **Ready**.  
If any backups are still running, wait for them to complete before proceeding to the next topic. The **Ready** status indicates that the backup was finished successfully. If the status is **Failed** then the backup did not complete successfully.

---

## Management subsystem: updating the password, keys, and secrets

Update the cloud administrator password, the key for encrypted database fields, and the **client\_ID** and **client\_secret** values for the Management subsystem in an API Connect deployment.

---

### About this task

- [Changing the cloud administrator password](#)  
Update the cloud administrator password for the Management subsystem using the Cloud Manager UI, the toolkit CLI, or an API call.
- [Changing the database encryption key](#)  
Replace the value of the secret that is used for encrypting database fields in the Management subsystem.
- [Changing the registration client id and client secret for applications](#)  
Change the registration credentials (the **client\_id** and **client\_secret**) for applications provided by the Management subsystem.

---

## Changing the cloud administrator password

Update the cloud administrator password for the Management subsystem using the Cloud Manager UI, the toolkit CLI, or an API call.

---

### About this task

In the Management subsystem for an API Connect on-premises deployment, the administrator is called the *cloud administrator* and works with the management server to configure the deployment of the local server cloud. Updating the cloud administrator password for the Management subsystem requires no service interruption because the change takes place immediately.

You can change the cloud administrator's password using any of the following methods.

Note: Passwords must contain eight or more characters, with at least one character from three of the following categories: lowercase letters, uppercase letters, numerals, and punctuation (for example: ! \$ # %). The same character cannot be used more than twice consecutively.

Cloud Manager UI

1. [Log in to the Cloud Manager UI.](#)
2. [Change your Cloud Manager password.](#)

Toolkit CLI

1. Run the `apic login` command to [log in to a management server](#).
2. Run the `apic me:change-password` command to [change your password](#).

API call

1. Run the `apic login` command to [log in to a management server](#).  
You must log in to a management server with the toolkit CLI before you can call the API Connect APIs.
2. Execute the `/api/me/change-password` API call, as explained in the "Resource: Me" section of the IBM API Connect Platform - Cloud Management API.

The next time that you log in as the cloud administrator, you must use the new password.

---

## Changing the database encryption key

Replace the value of the secret that is used for encrypting database fields in the Management subsystem.

---

### About this task

This update does not require an outage; however some operations might take longer than usual due to database locking.

---

### Procedure

1. Create a secret rotation CR that specifies the name of your management cluster.  
You can optionally specify the name of a secret containing the new `encryption_secret.bin` as shown in the following example, or you can omit it so that a new key is generated automatically.

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementSecretRotation
metadata:
 name: rotate-secret
spec:
 managementCluster: management
 rotateEncryptionSecret:
 rotate: true
 # encryptionSecret: name_of_secret
```

Note: If you upgraded from V2018, the database encryption key was carried forward and is called `managementUpgradeName-encryption-secret`.

2. Run the following command to apply the CR and change the secret:

```
kubectl -n namespace create -f rotate_secret.yaml
```

where:

- `namespace` is the namespace where you installed the Management subsystem.
  - `rotate_secret.yaml` is the name of the file containing the CR. Notice that the file name does not have to match the value of the `name` specified in the CR.
3. Verify that the secret was generated using one of the following methods:
    - Run the `kubectl get job` command and verify that the results include a job for the new CR.  
The job is named based on the management cluster name and the CR name, using the following format: `managementCluster-crName`. For the example CR `rotate-secret` created for the `management` cluster, the job is named `management-rotate-secret`.
    - Look in the pod logs of the `apiconnect` operator and locate the record with the name that you specified in the `kind` field in the CR, and check its progress.  
For the example CR, the `kind` value is "ManagementSecretRotation".  
You can also use the log to verify that database tables were updated.

---

## Changing the registration `client_id` and `client_secret` for applications

Change the registration credentials (the `client_id` and `client_secret`) for applications provided by the Management subsystem.

---

### About this task

Update the application registration `client_id` and `client_secret` by changing the Kubernetes secret containing them. The `apim` file-watcher detects the change and automatically updates the following secrets in the database:

- toolkit: `management-cli-cred` (Customer facing)  
When you update the secret for the toolkit, users must download and install new toolkit credentials as explained in [Installing the toolkit](#).
- consumer-toolkit: `management-ccli-cred` (Customer facing)  
When you update the secret for the toolkit, users must download and install new toolkit credentials as explained in [Installing the toolkit](#).

- designer: **management-dsgr-cred** (Customer facing)  
When you update the secret for API Designer, users must download and install new credentials as explained in [Installing the toolkit](#).
- atm: **management-atm-cred** (Internal)
- governance: **management-governance-cred** (Internal)
- consumer-ui: **management-cui-cred** (Internal)
- juhu: **management-juhu-cred** (Internal)
- ui: **management-ui-cred** (Internal)  
The API Manager and Cloud Manager read the credentials from the back-end on start-up.

When you update registration credentials, the registration name that is associated with the updated secret changes. Table 1 lists the registration secrets with the default registration name and the pattern used for generating updated registration names for each secret. The new registration names follow the patterns shown in the table. The second and subsequent updates append a number to the name; for example, **toolkit-application-registration**, **toolkit-application-registration0** and then **toolkit-application-registration1**.

Table 1. Secret names and the corresponding default, and new, registration names for each secret

Secret	Default registration name	New registration name pattern
<b>management-cli-cred</b>	<b>toolkit-default</b>	<b>toolkit-application-registration*</b>
<b>management-ccli-cred</b>	<b>consumer-toolkit-default</b>	<b>consumer-toolkit-application-registration*</b>
<b>management-dsgr-cred</b>	<b>designer-default</b>	<b>designer-registration*</b>
<b>management-atm-cred</b>	<b>atm-default</b>	<b>atm-default*</b>
<b>management-governance-cred</b>	<b>compliance-default</b>	<b>compliance-default*</b>
<b>management-cui-cred</b>	<b>consumer-ui-default</b>	<b>consumer-ui-default*</b>
<b>management-juhu-cred</b>	<b>juhu-default</b>	<b>juhu-default*</b>
<b>management-ui-cred</b>	<b>ui-default</b>	<b>ui-default*</b>

## Procedure

1. To renew an internal certificate, complete the following steps:
  - a. Create a ManagementSecretRotation CR (for example, operationcr.yaml) with contents similar to the following example, and list the secrets that you want to update:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementSecretRotation
metadata:
 name: rotate-credentials
spec:
 managementCluster: management
 rotateClientIDandSecret:
 secrets:
 - management-ui-cred
 - management-cui-cred
 - management-juhu-cred
 - management-atm-cred
```

- b. Apply the CR by running the following command:

```
kubectl create -f operationcr.yaml -n namespace
```

where:

- **namespace** is the namespace where you installed the Management subsystem
- **operationcr.yaml** is the file that contains the CR

Applying the resource updates the credential for the specified secrets with new values.

2. To update customer-facing credentials, complete the following steps:
  - a. For each of the credential secrets, run the following command to update the **client\_id** and **client\_secret**:

```
kubectledit secret credential_secret_name -n namespace
```

where:

- **namespace** is the namespace where the Management subsystem is installed.
- **credential\_secret\_name** is the name of the secret that you want to update.

The response opens the secret in a text editor as shown in the following example:

```
apiVersion: v1
data:
 credential.json:
 eyAiaWQiOiAiNTY3MTYyNjYtZDE0Zi00ODZmLWl0MmYtZTE3MzZkZTg3NjE0IiwgInNlY3JldCI6ICJmYjU5MDdhNy0zMjZjLTQ4ZWYtYjQ0ZC1hMTI5ZmZkMzNiMTEiIH0=
```

In the secret, the value of **credential.json** is a base64 encoded string of the following JSON format. Decode it first, modify the values of **id** and **secret**, and encode back as base64 and paste it in the secret. (Note that either **id** or **secret** or both can be updated, but if the credentials have been compromised then it's recommended to update both id and secret). For example:

- b. Decode the value of **credential.json**.

To decode the value, copy it from the secret in step 2a, and paste it between the quotation marks as the **encoded\_value** in the following command:

```
echo "encoded_value" | base64 --decode
```

Using the value returned in step 1, the command looks like the following example:

```
echo
"eyJhbnQiOiAiNTY3MTYyNjYtZDE0Zi00ODZmLWl0MmYtZTE3MzZkZTg3NjE0IiwgInNlY3JldCI6ICJmYjU5MDdhNy0zMjZjLTQ4ZWYtYjQ0ZC1hMTI5ZmZkMzNiMTEiIH0=" | base64 --decode
```

The response displays decoded values of the **id** and **secret**, as in the following example:



Update the administrator password and the `client_secret` values for the Gateway subsystem in an API Connect deployment.

## About this task

---

- [Changing the Gateway administrator password](#)  
Change the Gateway administrator password by updating the CR and reinstalling the subsystem.
- [Changing the Gateway client\\_secret](#)  
The Gateway subsystem receives its `client_secret` from the Management subsystem.
- [Changing the secret for Gateway peering](#)  
Update the secret that protects gateway-peering communications within a Gateway cluster in your API Connect deployment.
- [Updating the TLS profile for Gateway](#)  
Update the TLS profile that secures communications with the Gateway subsystem in your API Connect deployment.

---

## Changing the Gateway administrator password

Change the Gateway administrator password by updating the CR and reinstalling the subsystem.

### About this task

---

Updating the administrator password for the Gateway subsystem requires no measurable reduction in API enforcement performance (gateways remain "up"); however, services such as SNMP will experience a disconnect until the password is updated on all pods.

### Procedure

---

1. Determine the name of the secret that is used in the Gateway CR by running the following command:

```
kubectl -n namespace get gatewaycluster gw-cluster-name -o yaml
```

where `namespace` is the namespace where you installed the Gateway subsystem.

In the resource, the Gateway's `secretName` is in the `adminUser` section.

2. Update the secret by running the following command to edit the resource:

```
kubectl -n namespace edit secret secretName
```

where `namespace` is the namespace where you installed the Gateway subsystem.

Change the value of `.data.password` to be the base64-encoded value of a new password. The new password can be any string you want but you must encode it to base64.

3. Save and close.  
The Gateway pods automatically restart, one at a time, until all pods are using the new secret.
4. To verify that the Gateway subsystem(s) are fully installed, run the following command:  
Replace `namespace` with the appropriate namespace.

```
kubectl -n namespace get GatewayCluster
```

where `namespace` is the namespace where you installed the Gateway subsystem.

The update is complete when the `READY` status is `True`, and the `SUMMARY` reports that all services are online ( `2/2` ) for all the Gateway subsystems that were updated. Example:

NAME	READY	SUMMARY	VERSION	RECONCILED VERSION	AGE
gww5	True	2/2	10.0.0.0	10.0.0.0-1219	7m31s
gww6	True	2/2	10.0.0.0	10.0.0.0-1219	7m32s

---

## Changing the Gateway `client_secret`

The Gateway subsystem receives its `client_secret` from the Management subsystem.

### About this task

---

If you changed the `client_secret` on the Management subsystem, force a dynamic reconfiguration and reregistration (DRR) as explained in [How can I force re-population of the configuration data?](#) On completion, the Gateway subsystem uses the new `client_secret` in its communications with the Management subsystem.

---

## Changing the secret for Gateway peering

Update the secret that protects gateway-peering communications within a Gateway cluster in your API Connect deployment.

### About this task

---



You can update the secret for an appliance-based gateway or a Kubernetes-based gateway. This task causes a disruption in API transactions because the secret must be changed on all gateways at the same time, and all gateways must be restarted.

## Procedure

---

1. Appliance only: Manually update the Gateway Peering Identification Credential and Validation Credential objects.

Attention: If your Gateway is not deployed on an appliance, skip to step 2.

As soon as you begin this process, nodes will be out of sync. The nodes will remain out of sync until the entire process is complete. If the nodes do not recover from the out-of-sync condition, you might need to perform a DRR. For information about dynamic reregistration and reconfiguration and (DRR), see [Dynamically re-registering and reconfiguring a Gateway service in a Kubernetes deployment](#).

2. Kubernetes only: If your Gateway is deployed on Kubernetes, update the secret by completing the following steps.

- a. Determine the name of your Gateway cluster by running the following command:

```
kubectl -n namespace get gatewayCluster
```

where *namespace* is the namespace where the Gateway subsystem is deployed.

- b. Determine the name of the secret currently being used for gateway peering by running the following command:

```
kubectl -n namespace describe gatewayCluster gateway-cluster
```

where:

- *namespace* is the namespace where the Gateway subsystem is deployed.
- *gateway-cluster* is the name that you obtained in step 2a.

In the response, the name of the secret currently being used displays after the heading **Apic Gateway Peering TLS** as shown in the following example:

```
Spec:
 Admin User:
 Secret Name: datapower-admin-credentials
 API Debug Probe: true
 API Debug Probe Expiration Minutes: 60
 API Debug Probe Max Records: 1000
 Apic Gateway Peering TLS:
 Secret Name: gateway-peering
```

- c. Show the age of the secret currently being used by running the following command:

```
kubectl -n namespace get secret gateway-peering-secret
```

where:

- *namespace* is the namespace where the Gateway subsystem is deployed.
- *gateway-peering-secret* is the secret name that you obtained in step 2b.

- d. Delete the secret and then immediately delete all of the Gateway pods in the Gateway cluster by running the following two commands:

```
kubectl -n namespace delete secret gateway-peering-secret
kubectl -n namespace delete pod gw-pod1 gw-pod2 gw-pod3
```

where:

- *namespace* is the namespace where the Gateway subsystem is deployed.
- *gateway-peering-secret* is the secret name that you obtained in step 2b.

When you delete the secret and the pods, the following events occur automatically:

- i. The cert-manager recreates the secret.
- ii. The DataPower operator recreates the gateway pods.
- iii. Dynamic reconfiguration re-installs the API metadata onto the gateways.

For information about dynamic reregistration and reconfiguration and (DRR), see [Dynamically re-registering and reconfiguring a Gateway service in a Kubernetes deployment](#).

- e. To verify the update, show the age of the new gateway peering secret by running the following command:

```
kubectl -n namespace get secret gateway-peering-secret
```

where:

- *namespace* is the namespace where the Gateway subsystem is deployed.
- *gateway-peering-secret* is the secret name that you obtained in step 2b.

Make sure that the secret is new and does not show the same age as previously.

---

## Updating the TLS profile for Gateway

Update the TLS profile that secures communications with the Gateway subsystem in your API Connect deployment.

## Procedure

---

1. Appliance only: update the TLS profile by modifying the configuration of the REST management interface.

Attention: If your Gateway is not deployed on an appliance, skip to step 2.

For information on the REST management interface commands, see *REST management interface commands* in the appropriate version of the [DataPower documentation](#).

2. Kubernetes only: update the TLS profile using the Domain API, which is exposed in the GatewayCluster CRD.

The update requires a change the `rest-mgmt` service in the `default` domain. For information, see [Domain API](#) in the DataPower Operator documentation. For information on updating the domain definition, see [Domain configuration](#) in the same documentation.

Note: The Gateway's API call SNI endpoint already supports replacing customer-provided keys and certifications as explained in [Binding a TLS server profile to a gateway service](#).

---

## Portal subsystem: updating keys and secrets

Update the database passwords, keys, and the `client_secret` values for the Portal subsystem in an API Connect deployment.

---

### About this task

- [Changing the secret for Portal data and system tools](#)  
Replace the encryption key that generates the secret used for securing the database and system tools that are used by the Portal subsystem.
- [Changing the Portal client secret](#)  
Change the `client_secret` for the Portal subsystem to keep it in sync with the Management subsystem.

---

## Changing the secret for Portal data and system tools

Replace the encryption key that generates the secret used for securing the database and system tools that are used by the Portal subsystem.

---

### Before you begin

If you are using a stand-alone, single data center deployment, you can update the encryption key with the PortalSecretRotation CR, as explained in [Renewing the portal CA with the PortalSecretRotation CR](#). The PortalSecretRotation CR is not supported for a two data center configuration or a Cloud Pak for Integration deployment, so you must update the key manually as explained in the following steps.

---

### About this task

The Portal subsystem uses a database plus several system tools to manage storage, hot backups, and cluster synchronization. The following services are all secured with the secret that is generated by the Portal CR during installation:

- MySQL root account for each Portal site
- Csync2 cluster synchronization service

Updating the portal-encryption-secret will make the following changes to your deployment:

- Change the root MySQL password
- Generate a new, site-specific MySQL password for each site
- Generate a new csync2 cluster key
- Re-encrypt all `client_ids` and `client_secrets` in the portal database

If you need to change the secret, the update is applied to all of the listed services. You can change the secret by generating a new value and restarting pods to pick up the new value. In a two data center deployment, you must manually copy the new secret to the other site.

---

## Procedure

Generate a new secret and apply it to the deployment using the appropriate steps for your environment:

- Single data center  
Run the following command to generate a new secret and apply it to the deployment:

```
kubectl -n namespace patch secret portal-enc-key -p "{\"data\": {\"encryption_secret\": \"$(head -c72 /dev/urandom | base64 -w0 | base64 -w0)\"}}"
```

where *namespace* is the namespace where you installed the Portal subsystem.

The `portal-enc-key` was generated during installation and added to the CR as the `encryptionSecret` setting. When you generate a new secret, the value is updated automatically. If you saved the secret in a file after the [Portal installation](#) as recommended, be sure to update that file now.

- Two data centers  
a. Generate the new secret on DC1 by running the following command:

```
head -c72 /dev/urandom | base64 -w0 | base64 -w0 > secret.txt
```

The `portal-enc-key` was generated during installation and added to the CR as the `encryptionSecret` setting. When you generate a new secret, the value is updated automatically. If you saved the secret in a file after the [Portal installation](#) as recommended, be sure to update that file now.

- b. Open a terminal to each DC in the same working directory where you saved `secret.txt` (or, copy the file to somewhere you can open a terminal to each DC).
- c. In each terminal, verify that you can access `secret.txt` by typing the following command:

```
echo $(cat secret.txt)
```

- d. In each terminal, type the following command -- but do not press Enter yet:

```
kubectl -n namespace patch secret portal-enc-key -p "{\"data\": {\"encryption_secret\": \"$(cat secret.txt)\"}}"
```

where *namespace* is the namespace where you installed the Portal subsystem.

- e. Press Enter in both terminals at the same time (or as close in time as possible).

There is a 5-minute time-out that starts as soon as any of the pods has the new secret mounted by Kubernetes. All pods will get the secret at slightly different times, but within about a minute of each other. If the 5-minute time-out expires before all pods across both DCs have the new secret, then the pods will rotate the new secret in at different times, resulting in errors in the logs and some down time for various pods, until they manage to rotate the new secret into use and recover.

---

## Results

When pods restart, the new secret is used to secure the services. The `portal-www` admin logs display messages as the new secret is used to re-encrypt the database. The following example shows sample log messages when the updated secret is applied to the database:

```
[upgrade stdout] 2872 888ec0:8cb2cc:73add2 2020-08-06 13:16:21: NEW k8s encryption key detected on this system!
Reencrypting portal data
[upgrade stderr] 2954 888ec0:73add2:7fce1b 2020-08-06 13:16:21: refresh_encryption_key: Command Started:
/opt/ibm/bin/refresh_encryption_key
FEgX8F5MxtwvoJ8Wpxw5DB/6MKK/qPKfRt5FaId3jy5IjbpXL8wi3/33q3KOySh8/zh11rVNFZTlzRnVfkGheLr9Le/Pb9jq
[upgrade stdout] 2872 888ec0:8cb2cc:73add2 2020-08-06 13:16:21: No portal service exists! Simply overriding the key
[upgrade stdout] 2872 888ec0:8cb2cc:73add2 2020-08-06 13:16:21: New key installed
(FEgX8F5MxtwvoJ8Wpxw5DB/6MKK/qPKfRt5FaId3jy5IjbpXL8wi3/33q3KOySh8/zh11rVNFZTlzRnVfkGheLr9Le/Pb9jq)
[upgrade stderr] 2954 888ec0:73add2:7fce1b 2020-08-06 13:16:21: refresh_encryption_key: Command Finished:
/opt/ibm/bin/refresh_encryption_key
FEgX8F5MxtwvoJ8Wpxw5DB/6MKK/qPKfRt5FaId3jy5IjbpXL8wi3/33q3KOySh8/zh11rVNFZTlzRnVfkGheLr9Le/Pb9jq with RC 0
[upgrade stderr] 2872 888ec0:8cb2cc:73add2 2020-08-06 13:16:21: Command Finished: /opt/ibm/bin/sysupgrade with RC 0
```

If there is a Portal service registered and Developer Portal sites installed, then this task re-encrypts that data as well. When the site encryption is complete, the log displays a "Command Finished" message for the `refresh_encryption_key` command, as shown in the following example:

```
Command Finished: /opt/ibm/bin/refresh_encryption_key <KEY> with RC 0
```

---

## Changing the Portal `client_secret`

Change the `client_secret` for the Portal subsystem to keep it in sync with the Management subsystem.

---

### About this task

The Portal subsystem uses the same `client_secret` as the Management subsystem. If you change the `client_secret` on the Management subsystem, update the Portal subsystem by pulling the credential from the Management subsystem.

Note: If you did not deploy the Portal subsystem, skip this task.

---

### Procedure

1. On a Portal pod, run the following command to pull the Portal service's `client_secret` from the Management subsystem:

```
kubectl -n namespace exec -t portal-mydc-www-0 -- bash -ic refresh_cloud_credentials
```

where `namespace` is the namespace where you installed the Portal subsystem.

2. Then, run the following command to update the credential for the Portal sites:

Note: If you did not deploy any Portal sites, skip this step.

```
kubectl -n namespace exec -t portal-mydc-www-0 -- bash -ic "refresh_site_credentials -a"
```

where `namespace` is the namespace where you installed the Portal subsystem.

---

## Analytics subsystem: replacing the TLS profile and keys

Update the Analytics subsystem's TLS profile and keys for backup and restore operations in your API Connect deployment.

- [Updating the TLS profile for Analytics](#)  
Update the TLS Profile for the Analytics service in your API Connect deployment.
- [Updating the Analytics backup/restore secret](#)  
Update the secret that secures backed-up data for the Analytics subsystem in an API Connect deployment.

---

## Updating the TLS profile for Analytics

Update the TLS Profile for the Analytics service in your API Connect deployment.

---

### Procedure

1. Unassociate Analytics service from the Gateway service.  
Unassociate the Analytics service from all gateway services as explained in step 7 in the topic, [Associating an analytics service with a gateway service](#).
2. Update the TLS Profile for the Analytics service.

See step 3 in the topic, [Registering an analytics service](#).

3. Associate the Analytics service with the Gateway service.  
See [Associating an analytics service with a gateway service](#).

---

## Updating the Analytics backup/restore secret

Update the secret that secures backed-up data for the Analytics subsystem in an API Connect deployment.

### About this task

When you configure S3 storage for backing up Analytics data, you encrypt the S3 provider's access key and save it as a Kubernetes secret for your Analytics deployment. If the S3 access key or your Kubernetes secret is compromised, obtain a new key from the S3 provider and use it to generate a new Kubernetes secret.

Important: When you invalidate a key, be sure to save a copy of it with a notation listing the dates when it was used. If you need to restore a backup that was created with the invalidated key, you will need to use that key to access the data. One method of saving the key is to create a new version of the CR every time you change the key, so that you can easily refer to the settings and key that were used for each backup.

This task assumes that you have already [configured the backup settings for your Analytics subsystem](#) and now want to replace the Kubernetes secret that is specified as the `analytics-backup-secret` in the Analytics CR.

### Procedure

1. Update the secret for your S3 storage:
  - a. Obtain a new access key and corresponding access key secret from your S3 provider.
  - b. Invalidate the previous credentials that were specified in the Analytics CR.
2. Delete the invalidated Kubernetes secret by running the following command:

```
kubect1 -n namespace delete secret name
```

where:

- `namespace` is the namespace where Analytics is deployed.
- `name` is the name of the current (invalidated) Kubernetes secret, which is specified in the `credentials` setting in the `databaseBackup` section of the Analytics CR. Every time you update the secret, you must assign it a new name.

3. Create a new Kubernetes secret by running the following command and filling in your access key, access key secret, and namespace:

```
kubect1 -n namespace create secret generic new_secret_name --from-literal=access_key='S3_access_key' --from-literal=secret_key='S3_access_key_secret'
```

where:

- `namespace` is the namespace where Analytics is deployed.
- `new_secret_name` is a new name for storing the new Kubernetes secret. Every time you update the secret, you must assign it a new name.
- `S3_access_key` and `S3_access_key_secret` are the values that you received from the S3 provider.

4. Edit the Analytics CR file and replace the invalidated secret with the new secret.
  - a. In the CR, locate the `credentials` setting in the `databaseBackup` section.
  - b. Replace the invalidated secret name with the name of the new secret that you just generated.
  - c. Save and close the CR file.
5. Run the following command to apply the updated CR so that the new secret takes effect:

```
kubect1 -n namespace apply -f path/to/analytics-cr
```

where:

- `namespace` is the namespace where Analytics is deployed.
- `path/to/analytics-cr` is the path and file name of the Analytics CR.

---

## Renewing TLS certificates

Renew TLS certificates in your API Connect deployment.

A default API Connect deployment uses cert-manager to create *issuers*, *CA certificates*, *server certificates*, and *client certificates*. Cert-manager monitors all the certificates that it creates, and renews them before they expire.

Note: If you are not using cert-manager, or if you customized some or all of your API Connect certificates, then you are responsible for monitoring and renewing those certificates.

Certain API Connect configuration and maintenance operations can require that some or all TLS certificates that are used by API Connect are renewed before their expiry. The process of updating API Connect TLS certificates is called "certificate renewal". The process to renew certificates involves the following steps:

1. Manually deleting Kubernetes secrets that contain the TLS certificates that need renewal.
2. Cert-manager detecting the deletion of these secrets, and automatically creating new secrets that contain newly generated x509 certificates.

Steps steps for identifying and renewing API Connect certificates are described in the following topics:

- [Renewing certificates with cert-manager](#)  
Use cert-manager to renew the issuers, CA certificates, and derived certificates that it manages for your API Connect deployment.
- [Renewing custom certificates](#)  
If you did not use cert-manager to generate certificates for your API Connect deployment, you must renew them manually.

- [List of Issuers, CA certificates, and secrets](#)  
A summary of certificate issuers, with the CA issued by each, and the corresponding secret used to sign certificates in an API Connect deployment.
- [List of ingress certificates](#)  
A summary of certificates that are used for communications between subsystems and clients in an API Connect deployment.
- [List of intra-subsystem certificates](#)  
A summary of certificates used for communications within subsystems in an API Connect deployment.
- [V2018 upgrade: List of certificates to update manually](#)  
If you upgraded to API Connect V10 from V2018, some external certificates are carried over from the V2018 deployment and are not managed by cert-manager. When you update any of the retained certificates, you must restart the affected pods manually.
- [Renewing certificates on VMware](#)  
Replacing passwords, keys, and certificates on your VMware API Connect deployment.
- [Renewing certificates in a two data center deployment on Kubernetes and OpenShift](#)  
Renewing TLS certificates in an API Connect two data center disaster recovery deployment requires you to copy some updated information between data centers while ensuring that the shared information is not overwritten during the remaining updates.

## Related information

- [API Connect TLS certificates reference page.](#)

## Renewing certificates with cert-manager

Use cert-manager to renew the issuers, CA certificates, and derived certificates that it manages for your API Connect deployment.

Cert-manager monitors and automatically renews all API Connect certificates before they expire. However, some API Connect maintenance operations require the manual renewal of certificates. The topics in this section describe how to manually trigger cert-manager to renew the certificates that it manages.

Key points to understand:

- If you replaced any cert-manager certificates with custom certificates, you must update these certificates manually before they expire. Cert-manager does not monitor custom certificates.
- Some API Connect pods must be restarted when certain API Connect certificates are renewed. See [Pods that require restart after certificate renewal](#).
- When you renew a CA certificate, you must renew all the end-entity certificates that the CA signs. See [API Connect TLS certificates reference](#).

Note: If your API Connect deployment was originally installed at v10.0.1.2 or earlier, then the certificate might be missing the `duration` and `renewBefore` properties. Before you renew any API Connect connect certificates, verify that your certificates have these properties, and if not then add them:

1. Run the following command to list of all the certificate names, with their `duration` and `renewBefore` properties:

```
kubectl -n <namespace> get certificate -o custom-
columns=NAME:metadata.name,DURATION:spec.duration,RENEWBEFORE:spec.renewBefore
```

Example output where `management-ca` is missing the `duration` and `renewBefore` properties:

NAME	DURATION	RENEWBEFORE
analytics-ai-endpoint	17520h0m0s	720h0m0s
...		
management-ca	<none>	<none>

2. For each certificate output from step 1 that shows `<none>` for `duration` and `renewBefore`, edit the certificate and add these properties:

```
kubectl -n <namespace> edit certificate <certificate name>
```

Add the `duration` and `renewBefore` properties under the `spec` property as shown in the following example:

```
...
spec:
 ...
 duration: <duration>
 renewBefore: 720h # 30 days
 ...
```

where `<duration>` is:

- `87600h` for CA certificates.
  - `17520h` for all other certificates.
- [Renewing the management CA with the ManagementSecretRotation CR](#)  
Use the ManagementSecretRotation CR to renew the management CA and all end-entity certificates that the management CA signs.
  - [Renewing the portal CA with the PortalSecretRotation CR](#)  
Use the PortalSecretRotation CR to renew the portal CA and all end-entity certificates that the portal CA signs.
  - [Renewing the ingress-ca](#)  
Renew the ingress CA certificate, and all the end-entity certificates that the ingress CA signs.
  - [Renewing the analytics CA](#)  
Renew the analytics CA certificate, and all end-entity certificates that the analytics CA signs.
  - [Renewing end-entity certificates](#)  
Renew an API Connect end-entity certificate.
  - [Monitoring cert-manager certificate renewal](#)  
How to monitor when your certificates are nearing their expiry, confirm that cert-manager is renewing them, and verify that dependent certificates are also renewed and related pods are restarted.

## Renewing the management CA with the ManagementSecretRotation CR

Use the `ManagementSecretRotation` CR to renew the management CA and all end-entity certificates that the management CA signs.

## About this task

Applying the `ManagementSecretRotation` CR (Custom Resource) is the recommended method for renewing the `management-ca` and all its end-entity certificates. The alternative method is to manually renew the management CA certificate and then each of its end-entity certificates, which you can identify from the certificates table [Management certificates](#).

Restriction: The `ManagementSecretRotation` CR is for use with a single data center deployment. Do not attempt to use it with a two data center disaster recovery deployment. See [Renewing certificates in a two data center deployment on Kubernetes and OpenShift](#).

## Procedure

1. Create a file called `management_cert_rotate.yaml` and paste in the following contents:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementSecretRotation
metadata:
 name: mgmt-rotate-issuer
spec:
 managementCluster: <management CR name>
 rotateCertificates:
 certificates:
 - <management CR name>-ca
```

where `<management CR name>` is the name of your `ManagementCluster` CR. You can identify this name with:

```
kubectl get ManagementCluster -n <management namespace>
```

2. Apply the CR by running the following command:

```
kubectl create -f management_cert_rotate.yaml -n <management namespace>
```

Applying the CR updates the `management-ca`, and all end-entity certificates that `management-ca` signs. To view a list of the renewed certificates, run the following command:

```
kubectl describe ManagementSecretRotation mgmt-rotate-issuer -n <management namespace>
```

The `status` block in the command output shows the renewed certificates:

```
Status:
...
Phase: Completed
Rotated Certs:
 def-management-ca
 def-management-db-client-replicator
 def-management-db-client-primaryuser
 def-management-client
 def-management-db-client-postgres
 def-management-natscluster-mgmt
 def-management-sitel-postgres
 postgres-operator
 def-management-sitel-postgres-pgbouncer
 def-management-db-client-pgbouncer
 def-management-db-client-apicuser
 def-management-server
State: 1/1
...
```

3. When certificate rotation is successfully completed, delete the `ManagementSecretRotation` CR. Confirm that certificate rotation is finished with the command:

```
kubectl get ManagementSecretRotation mgmt-rotate-issuer -n <management namespace>
```

The output should show `Completed`:

NAME	READY	STATUS	AGE
mgmt-rotate-issuer	1/1	Completed	6m52s

Then delete the CR:

```
kubectl delete ManagementSecretRotation mgmt-rotate-issuer -n <management namespace>
```

## Renewing the portal CA with the PortalSecretRotation CR

Use the `PortalSecretRotation` CR to renew the portal CA and all end-entity certificates that the portal CA signs.

## About this task

Applying the `PortalSecretRotation` CR (Custom Resource) is the recommended method for renewing the portal CA certificates and all its end-entity certificates. The alternative method is to manually renew the portal CA certificate and then each of its end-entity certificates, which you can identify from the certificates table [Portal certificates](#).

Renewing the `portal-ca` certificate causes downtime because all the `portal-db` pods need to restart the MySQL process to allow for the certificate to update. The pods do not restart, but they go non-ready and then ready again, which takes a few minutes.

Restriction: The `PortalSecretRotation` CR is for use with a single data center deployment. Do not attempt to use it with a two data center disaster recovery deployment. See [Renewing certificates in a two data center deployment on Kubernetes and OpenShift](#).

## Procedure

1. Create a file called `portal-secret-cr.yaml` and paste in the following contents:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalSecretRotation
metadata:
 name: portal-rotate-secret
spec:
 portalCluster: <portal CR name>
 # List of certificates you want to rotate (Listing an issuer will rotate any certs issued by the issuer e.g listing just
 portal-ca will rotate portal-ca, portal-client and portal-server)
 rotateCertificates:
 certificates:
 - <portal CR name>-ca
 rotateEncryptionSecret:
 # Set to true if you want to rotate the encryption secret.
 rotate: true
 # Optional value to set the encryption secret to, if supplied secret exists in the env. If not supplied the rotated
 secret will be random.
 encryptionSecret: new-encryption-secret
```

where `<portal CR name>` is the name of your `PortalCluster` CR. You can identify this name with:

```
kubectl get PortalCluster -n <portal namespace>
```

If you set `rotate: true` in the `rotateEncryptionSecret` section, the `portal-encryption-secret` is updated and the following changes are made:

- Change the root MySQL password.
- Generate a new, site-specific MySQL password for each site.
- Reencrypt all `client_ids` and `client_secrets` in the portal database.

2. Apply the CR by running the following command:

```
kubectl create -f portal-secret-cr.yaml -n <portal namespace>
```

Applying the CR updates the `<portal CR name>-ca`, along with all end-entity certificates that it signs. To view a list of the updated certificates, run the following command:

```
kubectl describe PortalSecretRotation portal-rotate-secret -n <portal namespace>
```

The `status` block in the command output shows the renewed certificates:

```
Status:
...
Phase: Completed
Rotated Certs:
 def-portal-ca
 def-portal-server
 def-portal-client
State: Completed Portal Secret Rotation.
...
```

3. When the certificate rotation is finished, delete the `PortalSecretRotation` CR.

Confirm that certificate rotation is finished with the command:

```
kubectl get PortalSecretRotation portal-rotate-issuer -n <portal namespace>
```

The output should show `Completed`:

NAME	STATUS	MESSAGE	AGE
portal-rotate-secret	Completed	Completed Portal Secret Rotation.	7m50s

Then delete the CR:

```
kubectl delete PortalSecretRotation portal-rotate-issuer -n <portal namespace>
```

---

## Renewing the ingress-ca

Renew the ingress CA certificate, and all the end-entity certificates that the ingress CA signs.

### About this task

The ingress CA certificate signs all user-facing and inter-subsystem certificates. When you renew the ingress CA, you must renew all the end-entity certificates that the ingress CA signs.

For more information about API Connect certificates, see [API Connect TLS certificates](#).

## Procedure

1. Run the following command to renew the ingress CA certificate:

```
kubectl -n <management namespace> get certificate <ingress CA name> -o=jsonpath='{.spec.secretName}' | xargs kubectl -n <management namespace> delete secret
```

where *<ingress CA name>* is the name of the ingress CA certificate. On Kubernetes and OpenShift individual subsystem installations this name is `ingress-ca`. On Cloud Pak for Integration and OpenShift top-level CR installations, this name is `<apic instance name>-ingress-ca`.


2. Run the following command to renew all the end-entity certificates that the ingress CA signs:

```
kubectl get secrets -n <management namespace> -o custom-columns='NAME:.metadata.name,ISSUER:.metadata.annotations.cert-manager.io/issuer-name' --no-headers=true | grep ingress-issuer | awk '{ print $1 }' | xargs kubectl delete secret -n <management namespace>
```

3. If you have other subsystems that are in different namespaces from the management subsystem, then follow the steps in [Copying renewed ingress-ca to subsystems in different namespaces](#).

4. Verify analytics in the developer portal.

Due to a known issue, when the `ingress-ca` is renewed, it is possible that [Analytics in the Developer Portal](#) might stop working. If this happens, complete the following steps to ensure that certificate changes take effect:

- a. Log in to the Cloud Manager user interface.
- b. In the navigation list, click  Topology.
- c. Edit the Analytics service.
- d. On the Analytics page, edit the Summary field to force a change; for example, by adding a space to the end of a sentence.
- e. Click Save.

---

## Copying renewed ingress-ca to subsystems in different namespaces

When the ingress CA certificate on the management subsystem is renewed, you must copy the renewed certificate to any subsystems that are in a different namespace.

---

### Before you begin

Complete the steps in [Renewing the ingress-ca](#) to renew the ingress CA in your management subsystem namespace.

Note: If all your API Connect subsystems are in the same namespace, you do not need to follow this procedure.

---

### About this task

The ingress CA signs all the inter-subsystem communication certificates that are used between API Connect subsystems. If your subsystems are installed in different namespaces, then the ingress CA must be manually synchronized across all those namespaces.

These steps must be repeated for all namespaces where you have subsystems that are registered with your management subsystem.

For more information about API Connect certificates, see [API Connect TLS certificates](#).

---

### Procedure

1. Export the management namespace ingress CA secret to a YAML file called `ingress-ca.yaml`:

```
kubectl -n <management namespace> get secret ingress-ca -o yaml > ingress-ca.yaml
```

2. Edit `ingress-ca.yaml` and remove all `metadata` fields except for the secret name. The resulting file contents look like this:

```
apiVersion: v1
data:
 ca.crt: ...
 tls.crt: ...
 tls.key ...
kind: Secret
metadata:
 name: ingress-ca
type: kubernetes.io/tls
```

3. Apply the updated the `ingress-ca.yaml` in the namespace of the other subsystem:

```
kubectl -n <other subsystem namespace> apply -f ingress-ca.yaml
```

4. Verify that the ingress CA is now identical in both namespaces.

- a. Extract your `ingress-ca` secret to a file in your management namespace:

```
kubectl -n <management namespace> get secrets ingress-ca -o yaml | grep tls.crt | grep -v 'f:tls' | awk '{print $2}' | base64 -d > ingress.pem.mgmt
```

- b. Extract your `ingress-ca` secret to a file in your other subsystems' namespace:

```
kubectl -n <other subsystem namespace> get secrets ingress-ca -o yaml | grep tls.crt | grep -v 'f:tls' | awk '{print $2}' | base64 -d > ingress.pem.other
```

- c. Verify that the files that are created for each namespace are identical:

```
diff ingress.pem.mgmt ingress.pem.other
```

5. Run the following command to renew all the `ingress-ca` end-entity certificates in the other subsystems' namespace:

```
kubectl get secrets -n <other subsystem namespace> -o custom-columns='NAME:.metadata.name,ISSUER:.metadata.annotations.cert-manager.io/issuer-name' --no-headers=true | grep ingress-issuer | awk '{ print $1 }' | xargs kubectl delete secret -n <other subsystem namespace>
```



---

## Renewing the analytics CA

Renew the analytics CA certificate, and all end-entity certificates that the analytics CA signs.

### About this task

---

The pods that comprise the analytics subsystem communicate with each other using certificates that are signed by the analytics CA certificate. If you renew the analytics CA certificate, you must also renew the analytics end-entity certificates that the analytics CA signs.

For more information about API Connect certificates, see [API Connect TLS certificates](#).

### Procedure

---

1. Run the following command to renew the analytics CA certificate:

```
kubectl -n <analytics namespace> get certificate <analytics CA name> -o=jsonpath='{.spec.secretName}' | xargs kubectl -n <analytics namespace> delete secret
```

where *<analytics CA name>* is the name of the analytics CA certificate. On Kubernetes and OpenShift individual subsystem installations this name is `analytics-ca`. On Cloud Pak for Integration and OpenShift top-level CR installations, this name is `<apic instance name>-a7s-ca`.

2. Renew the end-entity certificates signed by the analytics CA. Run the following command for each certificate:

```
kubectl get certificate <analytics certificate> -o=jsonpath='{.spec.secretName}' | xargs kubectl delete secret
```

where *<analytics certificate>* is the name of the end entity certificate. The analytics end-entity certificates are:

- `analytics-client` or `a7s-client`.
- `analytics-server` or `a7s-server`.

On Cloud Pak for Integration and OpenShift top-level CR installations, the certificate name is prefixed with `<apic instance name>-`.

---

## Renewing end-entity certificates

Renew an API Connect end-entity certificate.

### About this task

---

To renew an end-entity certificate, delete the secret that corresponds to the certificate. Cert-manager detects the deletion of the secret and re-creates it with a new x509 certificate.

For more information about API Connect certificates, see [API Connect TLS certificates](#).

### Procedure

---

1. Get a list of all certificates in your API Connect namespace:

```
kubectl -n <namespace> get certificates
```

The output lists all certificates with their status, age, and the name of the corresponding secret. In this example, the certificate that is used by the API Manager UI is shown:

NAME	READY	SECRET	AGE
...			
<code>apim-endpoint</code>	<code>True</code>	<code>apim-endpoint</code>	<code>13d</code>
...			

By default the secret names are identical to the certificate names.

2. Delete the secret for the certificate that you want to renew:

```
kubectl -n <namespace> delete secret <secret name>
```

Cert-manager detects the deletion of the secret and creates a new secret with a newly generated x509 certificate.

---

## Monitoring cert-manager certificate renewal

How to monitor when your certificates are nearing their expiry, confirm that cert-manager is renewing them, and verify that dependent certificates are also renewed and related pods are restarted.

### CA certificate renewal

---

If cert-manager renews a CA certificate, you must renew all the certificates that the CA signs. See [API Connect TLS certificates reference](#).

## Monitoring when your certificates are due to expire

Check when your certificates are due to expire with the following command:

```
kubectl -n <namespace> get certificate -o custom-
columns=NAME:metadata.name,DURATION:spec.duration,RenewBEFORE:spec.renewBefore,NotAFTER:status.notAfter
```

Example output:

NAME	DURATION	RenewBEFORE	NotAFTER
analytics-ingestion-client	17520h0m0s	720h0m0s	2025-08-28T08:25:58Z
api-endpoint	17520h0m0s	720h0m0s	2025-08-28T08:28:40Z
apim-endpoint	17520h0m0s	720h0m0s	2025-08-28T08:28:21Z

The example output shows that your `analytics-ingestion-client` is due to expire at `2025-08-28T08:25:58Z`. The `RenewBEFORE` property shows that cert-manager should renew this certificate 720 hours before it expires.

## Monitoring the cert-manager manager logs

Monitor the cert-manager logs to confirm when certificates are renewed. Run the following command to display the cert-manager log:

```
kubectl logs <cert-manager pod name> -n <cert-manager namespace>
```

Example output:

```
I0108 14:23:18.001074 1 controller.go:129] cert-manager/controller/certificates "level"=0 "msg"="syncing item"
"key"="default/test-cert"
I0108 14:23:18.009940 1 issue.go:109] cert-manager/controller/certificates/certificates/issue "level"=0
"msg"="certificate issued" "related_resource_kind"="Secret" "related_resource_name"="test-cert"
"related_resource_namespace"="default"
I0108 14:23:18.018664 1 sync.go:331] cert-manager/controller/certificates/certificates "level"=0 "msg"="certificate
scheduled for renewal" "duration_until_renewal"="-5.018566857s" "related_resource_kind"="Secret" "related_resource_name"="test-
cert" "related_resource_namespace"="default"
I0108 14:23:18.019070 1 controller.go:135] cert-manager/controller/certificates "level"=0 "msg"="finished processing work
item" "key"="default/test-cert"
```

The 4 statements (shown in the example) indicate when a certificate is about to be renewed, and when the renewal is complete a few seconds later.  
Tip: Set up a logs-based alerting mechanism that informs you when cert-manager renews any certificate.

## Pods that require restart after certificate renewal

Some API Connect pods must be restarted when certificates that are used by the pod are renewed. The certificates that require pods to be restarted after renewal are:

- `analytics-ingestion-client` or `a7s-ing-client`. Restart `apim`, `taskmanager`, and `analytics-proxy` pods on the management subsystem.
- `gateway-client-client` or `gw-dr-client`. Restart the `apim` and `taskmanager` pods on the management subsystem.
- `analytics-ai-endpoint` or `a7s-ai-endpoint`. Restart the `mtls-gw` pod on the analytics subsystem.

## Renewing custom certificates

If you did not use cert-manager to generate certificates for your API Connect deployment, you must renew them manually.

## About this task

You can recreate custom certificates that you generated, or that were generated by the operator during installation. After you renew certificates, restart the affected pods to ensure that the updated certificates are used.

## Procedure

1. Renew the certificates that you generated:  
Use one of the following methods to renew the certificates that you created:
  - Update the secret
  - Delete and then recreate the secret
  - Update the CR to reference a different secret name
2. Renew the certificates that were generated by the operator:  
To renew the certificates generated by the operator, delete each certificate so that it can be recreated by the operator.
3. Restart the affected pods so they can pick up the renewed certificates.  
For information on which pods are associated by each certificate, see the following topics:
  - [List of ingress certificates](#)
  - [List of intra-subsystem certificates](#)For instructions on restarting pods, see the following topics:
  - [Restarting Analytics pods and data collection](#)
  - [Management subsystem: restarting pods](#)
  - [Gateway subsystem: restarting pods](#)
  - [Portal subsystem: restarting pods](#)

## List of Issuers, CA certificates, and secrets

A summary of certificate issuers, with the CA issued by each, and the corresponding secret used to sign certificates in an API Connect deployment.

Table 1 presents a list of the certificate issuers used in an API Connect deployment, with the CA issued by each, and the corresponding secret used to sign certificates. The `selfsigning-issuer` is used only for creating CAs, and does not have a corresponding certificate or secret of its own. If you have customized certificates, the certificates generated by the issuers in Table 1 use the `custom-certs-external.yaml` template.

Table 1. Issuers, CAs, and secrets

Issuer	CA certificate	Secret
<code>selfsigning-issuer</code>	N/A	N/A
<code>ingress-issuer</code>	<code>ingress-ca</code> issued by <code>selfsigning-issuer</code>	<code>ingress-ca</code> Attention: If you change the <code>ingress-ca</code> secret in a 2 data center deployment, you must change it to the same value in both data centers.
<code>analytics-ca</code>	<code>analytics-ca</code> issued by <code>selfsigning-issuer</code>	<code>analytics-ca</code>
<code>management-ca</code>	<code>management-ca</code> issued by <code>selfsigning-issuer</code>	<code>management-ca</code>
<code>portal-ca</code>	<code>portal-ca</code> issued by <code>selfsigning-issuer</code>	<code>portal-ca</code>

Note: On Cloud Pak for Integration, and OpenShift top-level CR deployments, some certificate names are contracted and prefixed with the `APIConnectCluster` instance name. For example, the certificate `managment-ca` is called `<apic instance name>-mgmt-ca`.

## List of ingress certificates

A summary of certificates that are used for communications between subsystems and clients in an API Connect deployment.

Note: On Cloud Pak for Integration, and OpenShift top-level CR deployments, some certificate names are contracted and prefixed with the `APIConnectCluster` instance name. For example, the certificate `managment-ca` is called `<apic instance name>-mgmt-ca`.

## User-facing certificates

Table 1 presents a list of user-facing certificates. The corresponding secrets use the same names. If you customized your certificates, the certificates in Table 1 use the `custom-certs-external.yaml` template.

Table 1. User-facing certificates

Certificate name	Issuer	Description
<code>api-endpoint</code> or <code>mgmt-platform-api</code>	<code>ingress-issuer</code>	Platform API endpoint server certificate. The certificate that is presented to callers of the platform REST API, and to the toolkit CLI.
<code>consumer-endpoint</code> or <code>mgmt-consumer-api</code>	<code>ingress-issuer</code>	Consumer API endpoint server certificate. The certificate presented that is presented to callers of the consumer REST API, and to the toolkit CLI.
<code>apim-endpoint</code> or <code>mgmt-api-manager</code>	<code>ingress-issuer</code>	API Manager UI server certificate.
<code>cm-endpoint</code> or <code>mgmt-admin</code>	<code>ingress-issuer</code>	Cloud Manager UI server certificate.
<code>portal-web</code>	<code>ingress-issuer</code>	The server certificate used on the <code>portalUIEndpoint</code> . It is the server certificate that is presented to portal site users, and which their browser authenticates. If there is a problem with this certificate, then users see an error message about an invalid or insecure certificate in their browser when they access a portal site.  The default <code>portal-web</code> certificate is issued by the <code>ingress-issuer</code> , which has a self-signed root certificate. Portal users might see a browser warning about use of a self-signed certificate.
<code>hub-endpoint</code>	<code>ingress-issuer</code>	Used by the Automated API behavior testing application. User-facing server certificate for the <code>hub-endpoint</code> . If there is a problem with this certificate, then the user's browser shows a warning or error message.
<code>turnstile-endpoint</code>	<code>ingress-issuer</code>	Used by the Automated API behavior testing application. User-facing server certificate for the <code>turnstile-endpoint</code> . If there is a problem with this certificate, then the user's browser shows a warning or error message.

## Inter-subsystem certificates

Table 2 lists all of the common subsystem communication certificates. The corresponding secrets use the same names. If you have customized your certificates, the certificates in Table 2 use the `custom-certs-external.yaml` template.

Table 2. Inter-subsystem certificates

Certificate name	Issuer	Description
------------------	--------	-------------

Certificate name	Issuer	Description
-ingress-ca	selfsigning-issuer	<p>The CA and issuer of all API Connect inter-subsystem and user-facing certificates. If there is a problem with this certificate, then all API Connect subsystems are inaccessible and unable to sync with each other.</p> <p>Update this certificate with kubectl. When updated, all child certificates must also be updated by deleting their corresponding secrets.</p> <p>The ingress-ca certificate is also stored in the management subsystem database. It is visible from the Cloud Manager UI, in the truststores of the analytics, gateway, event gateway, and portal default TLS client profiles. When the management subsystem <code>apim</code> pod is restarted the certificate is reloaded into the database from the Kubernetes ingress-ca certificate. Do not attempt to update this certificate from the Cloud Manager UI.</p> <p>In 2DCDR deployments, and when you have subsystems in different namespaces, it is necessary to manually copy the ingress-ca certificate from the management subsystem to the other subsystems. Steps are provided in the 2DCDR and multi-namespace install sections of this documentation.</p>
analytics-ingestion-client or a7s-ing-client	ingress-issuer	<p>Client certificate used for communication with the analytics subsystem on the ingestion endpoint. This certificate must have:</p> <ul style="list-style-type: none"> <li>The same CA as the server certificate <code>analytics-ai-endpoint</code> on the analytics subsystem.</li> <li>The common name must match the <code>spec.clientSubjectDN</code> in the analytics CR:</li> </ul> <pre>kubectl describe cert analytics-admin-client</pre> <pre>Spec:   Common Name: a7s-ing-client</pre> <pre>kubectl get ptl -o yaml</pre> <pre>spec:   clientSubjectDN: CN=a7s-ing-client</pre> <p>For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be <code>CN=a7s-ingestion-client</code>, or they could both be <code>CN=a7s-ingestion-client, O=cert-manager</code>, but they must be identical.</p> <p>To update this certificate, use kubectl. Restart the <code>apim</code>, <code>taskmanager</code>, and <code>analytics-proxy</code> pods after the update. This certificate is also used by the gateways for sending API events to the analytics subsystem. The updated certificate is sent by the management subsystem to the gateways, it is not necessary to restart any gateway pods.</p>
portal-admin-client or ptl-adm-client	ingress-issuer	<p>Client certificate used for communication with the portal subsystem on the <code>portalAdminEndpoint</code>. This certificate must have:</p> <ul style="list-style-type: none"> <li>The same CA as the server certificate <code>portal-admin</code> or <code>ptl-director</code> on the portal subsystem.</li> <li>The common name must match the <code>spec.adminClientSubjectDN</code> in the portal CR:</li> </ul> <pre>kubectl describe cert portal-admin-client</pre> <pre>Spec:   Common Name: ptl-adm-client</pre> <pre>kubectl get ptl -o yaml</pre> <pre>spec:   adminClientSubjectDN: CN=ptl-adm-client</pre> <p>For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be <code>CN=portal-admin-client</code>, or they could both be <code>CN=ptl-adm-client, O=cert-manager</code>, but they must be identical.</p>
gateway-client-client or gw-dr-client	ingress-issuer	<p>Client certificate for communications with the gateway service. Restart the <code>apim</code>, and <code>taskmanager</code> pods after update. This certificate must have the same CA as the <code>gwv6-manager-endpoint</code> or <code>gw-gateway-manager</code> certificate on the gateway.</p>
api-endpoint or mgmt-platform-api	ingress-issuer	<p>Platform API endpoint server certificate. The certificate that is presented to callers of the platform REST API, and to the toolkit CLI.</p>
consumer-endpoint or mgmt-consumer-api	ingress-issuer	<p>Consumer API endpoint server certificate. The certificate presented that is presented to callers of the consumer REST API, and to the toolkit CLI.</p>
apim-endpoint or mgmt-api-manager	ingress-issuer	<p>API Manager UI server certificate.</p>
cm-endpoint or mgmt-admin	ingress-issuer	<p>Cloud Manager UI server certificate.</p>
mgmt-replication-client	ingress-issuer	<p>2DCDR deployments only. Client certificate used in the warm-standby data center's <code>&lt;remote-sitename&gt;-postgres</code> pod to connect to the active data center.</p>
mgmt-replication-server	ingress-issuer	<p>2DCDR deployments only. Server certificate used in the active data center's <code>&lt;management_CR&gt;-tunnel</code> pod. This certificate must contain the DNS Subject Alternative Name of this data center's hostname.</p>
hub-endpoint	ingress-issuer	<p>Used by the Automated API behavior testing application. User-facing server certificate for the <code>hub-endpoint</code>. If there is a problem with this certificate, then the user's browser shows a warning or error message.</p>
turnstile-endpoint	ingress-issuer	<p>Used by the Automated API behavior testing application. User-facing server certificate for the <code>turnstile-endpoint</code>. If there is a problem with this certificate, then the user's browser shows a warning or error message.</p>
analytics-ai-endpoint or a7s-ai-endpoint	ingress-issuer	<p>Server certificate used on the analytics ingestion endpoint, in the <code>mtls-gw</code> pod. Management and gateway subsystems communicate with the analytics subsystem on this endpoint. The client certificates that are used by the management and gateway subsystems must use the same CA certificate as the analytics-ai-endpoint certificate.</p> <p>If this certificate is updated, restart the <code>mtls-gw</code> pod for the update to take effect.</p>

Certificate name	Issuer	Description
portal-admin or ptl-director	ingress-issuer	Server certificate used on the <code>portalAdminEndpoint</code> . The management subsystem communicates with the portal subsystem on this endpoint. The corresponding client certificate used in the management to portal communication is the <code>portal-admin-client</code> certificate.  This certificate must have the same CA as the <code>portal-admin-client</code> certificate on the management subsystem.
portal-web	ingress-issuer	The server certificate used on the <code>portalUIEndpoint</code> . It is the server certificate that is presented to portal site users, and which their browser authenticates. If there is a problem with this certificate, then users see an error message about an invalid or insecure certificate in their browser when they access a portal site.  The default <code>portal-web</code> certificate is issued by the <code>ingress-issuer</code> , which has a self-signed root certificate. Portal users might see a browser warning about use of a self-signed certificate.
ptl-replication-client	ingress-issuer	2DCDR deployments only. Client certificate that is used in the warm-standby data center's <code>&lt;remote-sitename&gt;-www</code> and <code>&lt;remote-sitename&gt;-db</code> pods to connect to the active data center.
ptl-replication-server	ingress-issuer	2DCDR deployments only. Server certificate used in the active data center's <code>&lt;portal_CR&gt;-tunnel</code> pod, the counterpart to the <code>ptl-replication-client</code> certificate. This certificate must contain the DNS Subject Alternative Name of this data center's hostname.
gateway-peering or gw-peer	ingress-issuer	This certificate secures the communication between gateways in your gateway cluster.
gmv6-endpoint or gw-gateway	ingress-issuer	
gmv6-manager-endpoint or gw-gateway-manager	ingress-issuer	The <code>gatewayManager</code> endpoint certificate. This is the server certificate on the gateway director endpoint, which the management subsystem communicates with. This certificate must be signed by the same CA as the <code>gateway-client-client</code> certificate on management subsystem
event-gateway-management-client	ingress-issuer	The <code>event-gateway-management-client</code> certificate exists only on Cloud Pak for Integration deployments.

## List of intra-subsystem certificates

A summary of certificates used for communications within subsystems in an API Connect deployment.

Note: On Cloud Pak for Integration, and OpenShift top-level CR deployments, some certificate names are contracted and prefixed with the `APIConnectCluster` instance name. For example, the certificate `management-ca` is called `<apic instance name>-mgmt-ca`.

Table 1 presents a list of certificates used for communications between pods in the same subsystem. The certificates are managed by cert-manager. For details of all the API Connect certificates, see [API Connect certificates](#).

Table 1. Intra-subsystem certificates

Certificate name	Issuer	Description
management-ca or mgmt-ca	selfsigning-issuer	The issuer for the management subsystems intra-subsystem certificates: management-client, management-server, postgres, and nats certificates. Communication between management subsystem pods fails if there is a problem with this certificate. This certificate is also used as the CA for REST API calls to the management subsystem from the other subsystems, when using <code>in-cluster</code> communication. See <a href="#">In-cluster service communication between subsystems</a>
management-client or mgmt-client	management-ca	Client certificate used in communication between management subsystem pods. Communication between management subsystem pods fails if there is a problem with this certificate.
management-server or mgmt-server	management-ca	Server certificate used in communication between management subsystem pods. Communication between management subsystem pods fails if there is a problem with this certificate. Required DNS names in the certificate:  * <code>&lt;namespace&gt;</code> * <code>&lt;namespace&gt;.svc</code> * <code>&lt;instance name&gt;-server.&lt;namespace&gt;.svc</code> <code>&lt;instance name&gt;-server</code>
db-client-apicuser	management-ca	Intra-subsystem certificate for the management database subsystem.
db-client-pgbouncer	management-ca	Intra-subsystem certificate for the management database subsystem.
db-client-postgres	management-ca	Intra-subsystem certificate for the management database subsystem.
db-client-primaryuser	management-ca	Intra-subsystem certificate for the management database subsystem.
postgres	management-ca	Intra-subsystem certificate for the management database subsystem.
postgres-pgbouncer	management-ca	Intra-subsystem certificate for the management database subsystem.
postgres-operator	management-ca	Intra-subsystem certificate for the management database subsystem.
natscluster-mgmt	management-ca	Intra-subsystem certificate for the <code>nats</code> pods.
db-client-replicator	management-ca	2DCDR deployments only. Client certificate used by the <code>&lt;management_CR&gt;-tunnel</code> pod to connect to the other data center's <code>&lt;management_CR&gt;-tunnel</code> pod.
analytics-ca or a7s-ca	selfsigning-issuer	The issuer for the analytics-client and analytics-server certificates. Communication between analytics subsystem pods fails if there is a problem with this certificate.

Certificate name	Issuer	Description
analytics-client or a7s-client	analytics-ca	Client certificate used in communication between analytics subsystem pods. Communication between analytics subsystem pods fails if there is a problem with this certificate.
analytics-server or a7s-server	analytics-ca	Server certificate used in communication between analytics subsystem pods. Communication between analytics subsystem pods fails if there is a problem with this certificate. Required DNS names in the certificate: <pre>*.&lt;namespace&gt; *.&lt;namespace&gt;.svc *.&lt;instance name&gt;-server.&lt;namespace&gt;.svc &lt;instance name&gt;-server &lt;instance name&gt;-storage</pre>
portal-ca or ptl-ca	selfsigning-issuer	The issuer for the portal-client and portal-server certificates. Communication between portal subsystem pods fails if there is a problem with this certificate.
portal-client or ptl-client	portal-ca	Client certificate used in communication between portal subsystem pods. Communication between portal subsystem pods fails if there is a problem with this certificate.
portal-server or ptl-server	portal-ca	Server certificate used in communication between portal subsystem pods. Communication between portal subsystem pods fails if there is a problem with this certificate. Required DNS names in the certificate: <pre>*.&lt;namespace&gt; *.&lt;namespace&gt;.svc *.&lt;instance name&gt;-server.&lt;namespace&gt;.svc &lt;instance name&gt;-server *.&lt;instance name&gt;-&lt;site name&gt;-db-all.&lt;namespace&gt;.svc *.&lt;instance name&gt;-&lt;site name&gt;-www-all.&lt;namespace&gt;.svc *.&lt;instance name&gt;-&lt;site name&gt;-db-all.&lt;namespace&gt;.svc.cluster.local *.&lt;instance name&gt;-&lt;site name&gt;-www-all.&lt;namespace&gt;.svc.cluster.local *.&lt;namespace&gt;.svc.cluster.local &lt;instance name&gt;-db #&lt;remote portal CR name&gt;-db # For 2DCDR only.</pre> <i>&lt;instance name&gt;</i> and <i>&lt;remote portal CR name&gt;</i> are truncated if more than 15 characters.

## V2018 upgrade: List of certificates to update manually

If you upgraded to API Connect V10 from V2018, some external certificates are carried over from the V2018 deployment and are not managed by cert-manager. When you update any of the retained certificates, you must restart the affected pods manually.

## Ingress (front-end) certificates

Table 1 presents a list of the ingress certificates that were carried forward when you upgraded from V2018, and the pods that you must restart when you update each certificate.

If multiple Portal web ingress endpoints have been configured, then the certificates for those additional endpoints must be updated manually as well.

Table 1. V2018 ingress certificates and affected pods

Certificate and Secret	Pods
client-tls-xxxxxxxxxx	<ul style="list-style-type: none"> <li>analytics-mtls-gw-hashed-suffix</li> </ul>
ingestion-tls-xxxxxxxxxx	<ul style="list-style-type: none"> <li>analytics-mtls-gw-hashed-suffix</li> </ul>
platform-api-tls-xxxxxxxxxx	N/A
api-manager-tls-xxxxxxxxxx	N/A
cloud-manager-tls-xxxxxxx	N/A
consumer-api-tls-xxxxxxxxxx	N/A
v5GwName-datapower-admin-credentials	N/A
v5GwName-apic-gateway-peering-tls	N/A
v5GwName-apic-gateway-service-tls	N/A
v6GwName-datapower-admin-credentials	N/A
v6GwName-apic-gateway-peering-tls	N/A
v6GwName-apic-gateway-service-tls	N/A
portal-uuid-admin	<ul style="list-style-type: none"> <li>portal-nginx</li> </ul>
www-tls-xxxxxxxxxxxxxxxxxx	N/A

## Subsystem communication certificates

Table 2 presents a list of the subsystem communication certificates that were carried forward when you upgraded from V2018, and the pods that you must restart when you update each certificate.

Table 2. V2018 subsystem communication certificates and affected pods

Usage	Certificate and Secret	Pods
-------	------------------------	------

Usage	Certificate and Secret	Pods
client	<code>managementUpgradeName-analytics-ingestion-client</code>	<ul style="list-style-type: none"> <li>• <code>management-apim</code></li> <li>• <code>management-taskmanager</code></li> <li>• <code>gateway</code> (via webhook, no restart needed)</li> </ul>
client	<code>managementUpgradeName-portal-admin</code>	<ul style="list-style-type: none"> <li>• <code>management-apim</code></li> <li>• <code>management-taskmanager</code></li> <li>• <code>management-portal-proxy</code></li> </ul>
client	<code>gw-apic-gateway-service-tls</code>	<ul style="list-style-type: none"> <li>• <code>gwv6</code></li> <li>• <code>gwv5</code></li> </ul>
clientserver	<code>gw-apic-gateway-peering-tls</code>	<ul style="list-style-type: none"> <li>• <code>gwv6</code> (redis)</li> <li>• <code>gwv5</code> (redis)</li> </ul>

## Database encryption key

The database encryption key is carried forward from V2018, and uses the name: `managementUpgradeName-encryption-secret`. You can update the value of this secret as explained in [Changing the database encryption key](#).

## Renewing certificates on VMware

Replacing passwords, keys, and certificates on your VMware API Connect deployment.

Attention:

The instructions in this section apply only to the management, portal, and analytics subsystems. For gateway appliance-specific instructions, see *Configuring the API Connect Gateway Service* in the appropriate version of the [DataPower documentation](#).

API Connect certificates on VMware are categorized as follows:

- **Common** certificates are used in communication between subsystems. Common certificates include CA and end-entity certificates. If you renew the common CA certificates `ingress-ca` and `root-ca`, then you must renew them on all subsystems, and also renew all the other common certificates.
- **Subsystem** certificates are specific to each subsystem. They all have the common `ingress-ca` certificate as their CA. If you renew `ingress-ca`, then you must renew all common and subsystem certificates on all subsystems.
- **Internal** certificates are used for communication between processes within a single subsystem. Internal certificates are renewed automatically and are not user-editable. Internal certificates are not listed in the `apicup certs list` output.

The common and subsystem certificates are the only certificates that you can modify, and can be listed with the `apicup certs list` command:

```
apicup certs list <subsystem>
```

Example output from a management subsystem:

```
Common certificates
```

```
=====
```

Name	Summary	Validation errors
<code>analytics-client-client</code>	CN: analytics-client-client SubjectKeyId: A3:6E:E8:6F:9F:F2:BD:3B:47:6D:E2:22:D2:28:D6:E7 AuthorityKeyId: 68:EF:43:39:72:49:15:DA:71:72:59:DC:80:2A:01:B8	
<code>analytics-ingestion-client</code>	CN: analytics-ingestion-client SubjectKeyId: BC:70:E0:A0:AB:96:D2:21:47:3C:DD:EB:6F:5B:81:5B AuthorityKeyId: 68:EF:43:39:72:49:15:DA:71:72:59:DC:80:2A:01:B8	
<code>ingress-ca</code>	CN: ingress-ca SubjectKeyId: 68:EF:43:39:72:49:15:DA:71:72:59:DC:80:2A:01:B8 AuthorityKeyId: 2F:75:CA:68:75:F0:51:AE:00:08:A2:4A:C4:AD:0D:DA	
<code>portal-client</code>	CN: portal-client SubjectKeyId: 18:D0:55:6A:74:64:CB:BA:AD:51:04:60:1D:CC:E3:49 AuthorityKeyId: 68:EF:43:39:72:49:15:DA:71:72:59:DC:80:2A:01:B8	
<code>root-ca</code>	CN: root-ca SubjectKeyId: 2F:75:CA:68:75:F0:51:AE:00:08:A2:4A:C4:AD:0D:DA AuthorityKeyId:	

```
Subsystem abc-management certificates
```

```
=====
```

Name	Summary	Validation errors
<code>api-manager-ui</code>	CN: api-manager-ui SubjectKeyId: DD:BA:15:D9:15:E1:86:8A:73:00:9F:42:63:53:70:D0 AuthorityKeyId: 68:EF:43:39:72:49:15:DA:71:72:59:DC:80:2A:01:B8	
<code>appliance-client</code>	CN: appliance-client SubjectKeyId: E6:FC:AC:64:2C:E9:9B:09:B1:1D:A8:05:12:EB:EB:56 AuthorityKeyId: BA:54:7D:C1:AC:73:A9:D9:26:0D:CC:02:1F:F6:A7:C0 D4:1D:8C:D9:8F:00:B2:04:E9:80:09:98:EC:F8:42:7E	
<code>atm-credential</code>	CN: cloud-admin-ui SubjectKeyId: E7:6A:82:4C:54:63:E8:4A:13:CF:81:A5:CB:97:A1:76 AuthorityKeyId: 68:EF:43:39:72:49:15:DA:71:72:59:DC:80:2A:01:B8	
<code>consumer-api</code>	CN: consumer-api SubjectKeyId: 9D:CF:2E:F8:22:85:8E:95:1E:CA:4B:C2:61:81:5F:BB AuthorityKeyId: 68:EF:43:39:72:49:15:DA:71:72:59:DC:80:2A:01:B8	
<code>consumer-toolkit-credential</code>	D4:1D:8C:D9:8F:00:B2:04:E9:80:09:98:EC:F8:42:7E	
<code>consumer-ui-credential</code>	D4:1D:8C:D9:8F:00:B2:04:E9:80:09:98:EC:F8:42:7E	

```

database-backup-auth-ssh-key D4:1D:8C:D9:8F:00:B2:04:E9:80:09:98:EC:F8:42:7E
designer-credential D4:1D:8C:D9:8F:00:B2:04:E9:80:09:98:EC:F8:42:7E
encryption-secret 21:50:5C:06:D7:17:09:A8:04:7C:D0:3A:61:D5:E3:6E
hub CN: hub
SubjectKeyId: 98:C5:80:BA:C5:3A:DB:27:A6:9F:92:7E:AD:38:6F:E8
AuthorityKeyId: 68:EF:43:39:72:49:15:DA:71:72:59:DC:80:2A:01:B8
juhu-credential D4:1D:8C:D9:8F:00:B2:04:E9:80:09:98:EC:F8:42:7E
k8s-ca CN: k8s-ca
SubjectKeyId: BA:54:7D:C1:AC:73:A9:D9:26:0D:CC:02:1F:F6:A7:C0
AuthorityKeyId:
management-replication-client CN: management-replication-client
SubjectKeyId: 4C:39:F2:F0:D7:78:F1:3D:68:ED:A4:35:F8:C5:2D:A9
AuthorityKeyId: 68:EF:43:39:72:49:15:DA:71:72:59:DC:80:2A:01:B8
management-replication-ingress CN: management-replication-ingress
SubjectKeyId: 99:55:65:B3:FF:60:4C:7E:52:01:C2:02:86:91:A8:96
AuthorityKeyId: 68:EF:43:39:72:49:15:DA:71:72:59:DC:80:2A:01:B8
platform-api CN: platform-api
SubjectKeyId: 9C:15:FE:C0:32:F2:AA:CF:D1:B6:C0:82:A5:69:F2:40
AuthorityKeyId: 68:EF:43:39:72:49:15:DA:71:72:59:DC:80:2A:01:B8
D4:1D:8C:D9:8F:00:B2:04:E9:80:09:98:EC:F8:42:7E
turnstile CN: turnstile
SubjectKeyId: D0:F0:0A:B2:45:23:2B:F5:09:75:1E:03:1C:BD:89:3E
AuthorityKeyId: 68:EF:43:39:72:49:15:DA:71:72:59:DC:80:2A:01:B8
D4:1D:8C:D9:8F:00:B2:04:E9:80:09:98:EC:F8:42:7E
v2018-mgmt-ca CN: v2018-mgmt-ca
SubjectKeyId: 3C:46:F6:61:A0:AC:85:F9:99:93:7F:E1:39:71:7C:77
AuthorityKeyId: 2F:75:CA:68:75:F0:51:AE:00:08:A2:4A:C4:AD:0D:DA
v2018-service-client CN: v2018-service-client
SubjectKeyId: D7:3C:5E:BA:B1:13:66:F5:25:46:2F:38:34:7B:34:99
AuthorityKeyId: 3C:46:F6:61:A0:AC:85:F9:99:93:7F:E1:39:71:7C:77
v2018-service-server CN: v2018-service-server
SubjectKeyId: 3D:EB:8A:D0:B4:65:16:3A:FF:B6:A0:00:F5:47:F5:63
AuthorityKeyId: 3C:46:F6:61:A0:AC:85:F9:99:93:7F:E1:39:71:7C:77

```

## Renewing certificates with the apicup command

Use the `apicup` command to renew certificates.

1. Clear the existing certificate with the `certs clear` operation:

```
apicup certs set --clear <subsystem> <certificate name>
```

2. Generate a replacement certificate for the one cleared in step 1:

```
apicup certs generate <subsystem>
```

3. Apply the new certificate to the subsystem with:

```
apicup subsys install <subsystem>
```

## Renewing intra-subsystem certificates on VMware

Intra-subsystem certificates are renewed automatically, and do not require any user action to update or renew.

## Using SSH to connect to the server so you can run Kubernetes commands

If the instructions in a referenced topic require you to run `kubect1` commands directly on servers, you must first log in to the virtual machine (appliance) as follows:

1. Run the following command to connect as the API Connect administrator, replacing `ip_address` with the appropriate IP address:

```
ssh ip_address -l apicadm
```

2. When prompted, select Yes to continue connecting.
3. When you are connected, run the following command to receive the necessary permissions for working directly on the appliance:

```
sudo -i
```

Then, you can follow the instructions in each topic and run `kubect1` commands as needed.

## Determining which certificates are due for renewal

To determine which certificates are due to expire and must be renewed, SSH into the server as explained in the previous section. Then, run the following command and check the `EXPIRATION` column in the results:

```
kubect1 get certificate
```

- [VMware: Renewing external certificates with apicup](#)  
On VMware OVA/Appliance, the external public-facing and the cross-subsystem certificates are managed with the `apicup` utility.

## VMware: Renewing external certificates with apicup

On VMware OVA/Appliance, the external public-facing and the cross-subsystem certificates are managed with the `apicup` utility.



## About this task

When API Connect is deployed on VMware, the public-facing external certificates, as well as some internal cross-subsystem certificates, are managed with **apicup** but are stored as Kubernetes secrets. Complete the following steps to renew certificates that are managed with **apicup** and restart the affected pods.

## Procedure

1. Renew the certificates as explained in [Replacing custom certificates](#).  
For information on setting up new certificates, see [Setting custom certificates](#).
2. Use Tables 1 and 2 to determine which certificates to renew and which pods to restart:  
Table 1 presents a list of secrets for external (ingress/front-end certificates) with the corresponding pod that must be restarted when the secret changes.

Table 1. External (ingress/front-end) secrets and affected pods

Secret	Pods
<b>analytics-ingestion-ingress</b>	<b>analytics-mtls-gw</b>
<b>platform-api</b>	N/A
<b>api-manager-ui</b>	N/A
<b>cloud-admin-ui</b>	N/A
<b>consumer-api</b>	N/A
<b>hub</b>	N/A
<b>portal-admin-ingress</b>	<b>portal-nginx</b>
<b>portal-www-ingress</b>	N/A
<b>turnstile</b>	N/A

Table 2 presents a list of secrets for internal (cross-subsystem) certificates with the corresponding pod that must be restarted when the secret changes.

Table 2. Internal (subsystem) secrets and affected pods

Secret	Pods
<b>analytics-client-client</b>	This is a legacy certificate, it is not used from v10.0.5 onwards.
<b>analytics-ingestion-client</b>	<ul style="list-style-type: none"><li>• <b>management-apim</b></li><li>• <b>management-taskmanager</b></li><li>• <b>gateway</b> (via webhook, no restart needed)</li></ul>
<b>portal-client</b>	<ul style="list-style-type: none"><li>• <b>management-apim</b></li><li>• <b>management-taskmanager</b></li><li>• <b>management-portal-proxy</b></li></ul>
Site-dependent names. Example: <b>management-replication-ingress/dc2-mgmt-replication</b>	<ul style="list-style-type: none"><li>• <b>management-tunnel</b></li></ul>
Site-dependent names. Example: <b>management-replication-client/dc2-mgmt-replication-client</b>	<ul style="list-style-type: none"><li>• <b>management-remote-sitename-postgres</b> (on warm-standby site in 2DC-HA config)</li></ul>
Site-dependent names. Example: <b>portal-replication-ingress/dc2-ptl-replication</b>	<ul style="list-style-type: none"><li>• <b>portal-tunnel</b></li></ul>
Site-dependent names. Example: <b>portal-replication-client/dc2-ptl-replication-client</b>	<ul style="list-style-type: none"><li>• <b>portal-remote-sitename-db-X</b></li><li>• <b>portal-remote-sitename-www-X</b></li></ul>

3. Restart the pods listed in the corresponding row for each certificate that you renewed.  
For instructions for restarting pods, see the following topics:
  - [Management subsystem: restarting pods](#)
  - [Gateway subsystem: restarting pods](#)
  - [Portal subsystem: restarting pods](#)
  - [Analytics subsystem: restarting endpoints and pods](#)

## Renewing certificates in a two data center deployment on Kubernetes and OpenShift

Renewing TLS certificates in an API Connect two data center disaster recovery deployment requires you to copy some updated information between data centers while ensuring that the shared information is not overwritten during the remaining updates.

## About this task

Begin the process by updating certificates as needed on data center 1 (DC1). Then, copy the **ingress-ca** from DC1 to data center 2 (DC2) before updating certificates on DC2. Finally, you will copy encryption keys for the Management and Portal subsystem from DC1 to DC2.

## Procedure

Complete the following steps in the sequence shown to ensure that you do not overwrite information that was copied from DC1 while updating DC2.

1. DC1: Renew TLS certificates as explained in [Renewing TLS certificates](#).
2. DC2: Complete the following steps to copy the **ingress-ca** from DC1:
  - a. On DC1, export the **ingress-ca** secret to a YAML file by running the following command:

```
kubectl -n namespace get secret ingress-ca -o yaml > ingress-ca.yaml
```

- b. Still on DC1, edit the `ingress-ca.yaml` file to remove all labels, annotations, `creationTimestamp`, `resourceVersion`, `uid`, and `selfLink`.
- c. Copy the `ingress-ca.yaml` to DC2.
- d. On DC2, apply the `ingress-ca.yaml` file by running the following command:

```
kubectl -n namespace apply -f ingress-ca.yaml
```

- e. Validate that the `ingress-ca` on DC2 matches the `ingress-ca` on DC1:
  - i. On DC1, run the following command to save the original `ingress-ca` as a PEM file:

```
kubectl -n namespace get secrets ingress-ca -o yaml | grep tls.crt | awk '{print $2}' | base64 -d > /tmp/ingress.pem.dc1
```

- ii. On DC2, run the following command to save the copied `ingress-ca` as a PEM file::

```
kubectl -n namespace get secrets ingress-ca -o yaml | grep tls.crt | awk '{print $2}' | base64 -d > /tmp/ingress.pem.dc2
```

- iii. Copy `/tmp/ingress.pem.dc1` from DC1 and store it in the `/tmp` folder on DC2.
- iv. On DC2, run the following command to compare the two files and verify that there are no differences:

```
diff /tmp/ingress.pem.dc1 /tmp/ingress.pem.dc2
```

3. DC2: Update all external certificates that are based on the `ingress-ca` (see [List of ingress certificates](#)).
4. DC2: Update all of the internal certificates (see [List of intra-subsystem certificates](#)).

At this point, DC1 and DC2 have the same `ingress-ca` secrets. Next, copy the encryption keys for the Management and Portal subsystems from DC1 to DC2.

5. Copy the encryption key for the Management subsystem to DC2 by completing the following steps:
  - a. On DC1, copy the management encryption key:

- i. Get the name of the management encryption key by running the following command:

```
$ kubectl -n namespace get mgmt -o yaml | grep enc
```

The response looks like the following example:

```
encryptionSecret: dallas-enc-key
```

Make a note of the secret name for the next step. In the example, the key's name is `dallas-enc-key`.

- ii. Retrieve the secret from the encryption key by running the following command (substitute in the name of the key from the previous step):

```
kubectl get secret dallas-enc-key -o yaml
```

The output looks like the following example:

```
apiVersion: v1
data:
 encryption_secret.bin:
VKBNFj7sAOizxvE1H6i+9P31oJHvWwsO+x*****EGe/K+x6b3D7FEWGgoyG1WBUJKB4+T21My2iR5rBTov
pyLiY5g*****tSiRcQKegMPNBPGl829SVBCxuv3I=
kind: Secret
metadata:
 creationTimestamp: "2020-08-28T12:53:41Z"
 labels:
 app.kubernetes.io/instance: m1
 app.kubernetes.io/managed-by: ibm-apiconnect
 app.kubernetes.io/name: dallas-enc-key
 name: dallas-enc-key
 namespace: default
 resourceVersion: "43039"
 selfLink: /api/v1/namespaces/default/secrets/m1-enc-key
 uid: 46c92395-9cc2-4421-b2c4-48e472c0cbb1
type: Opaque
```

- iii. Copy the output and save it in a YAML file; for example `dc1-enc-key.yaml`.
- iv. Edit the file and make the following changes:
  - Delete the contents of the `metadata`: section (retain the section).
  - In the `metadata`: section, add in a `name`: attribute, with `dc1-enc-key` as the value.

The following example shows the updated file:

```
apiVersion: v1
data:
 encryption_secret.bin:
VKBNFj7sAOizxvE1H6i+9P31oJHvWwsO+x*****EGe/K+x6b3D7FEWGgoyG1WBUJKB4+T21My2iR5rBTov
pyLiY5g*****tSiRcQKegMPNBPGl829SVBCxuv3I=
kind: Secret
metadata:
 name: dc1-enc-key
type: Opaque
```

- v. Copy the updated file (`dc1-enc-key.yaml`) to the `/tmp` folder on DC2.
- b. On DC2, run the following command to create a secret from the encryption key file (`dc1-enc-key.yaml`) that you copied from DC1:

```
$ kubectl -n namespace create -f /tmp/dc1-enc-key.yaml
```

The response looks like the following example:

```
secret/dc1-enc-key created
```

6. Update the encryption key for the Portal subsystem on DC1 and then copy it to DC2 as explained in [Changing the secret for Portal data and system tools](#).

## Management subsystem: restarting pods

Restart Management subsystem pods in your API Connect deployment.

### About this task

This section describes how to restart Management subsystem pods.

### Procedure

1. Determine the deployment name by running the following command:

```
kubectl -n namespace get deployment | grep CR_name
```

where:

- *namespace* is the namespace where the Management subsystem is installed.
  - *CR-name* is the name of the CR used for installing the Management subsystem.
2. Restart pods by running the appropriate `kubectl` commands, shown in Table 1.  
Use the deployment name that you obtained in step 1.

Table 1. Management pods and the commands to restart them

Pod	Restart command
management-analytics-proxy	<code>kubectl rollout restart deployment/<i>deployment_name</i> -n <i>&lt;namespace&gt;</i></code>
management-apim	<code>kubectl rollout restart deployment/<i>deployment_name</i> -n <i>&lt;namespace&gt;</i></code> A manual restart is not needed when the certificate is updated.
management-client-downloads-server	<code>kubectl rollout restart deployment/<i>deployment_name</i> -n <i>&lt;namespace&gt;</i></code> A manual restart is not needed when the certificate is updated.
management-hub	<code>kubectl rollout restart deployment/<i>deployment_name</i> -n <i>&lt;namespace&gt;</i></code>
management-juhu	<code>kubectl rollout restart deployment/<i>deployment_name</i> -n <i>&lt;namespace&gt;</i></code>
management-ldap	<code>kubectl rollout restart deployment/<i>deployment_name</i> -n <i>&lt;namespace&gt;</i></code> A manual restart is not needed when the certificate is updated.
management-lur	<code>kubectl rollout restart deployment/<i>deployment_name</i> -n <i>&lt;namespace&gt;</i></code> A manual restart is not needed when the certificate is updated.
management-natscluster	Execute for each <code>natscluster</code> pod, one at a time: <code>kubectl exec <i>management-natscluster-n</i> -- bash -c "kill -HUP 1"</code> A manual restart is not needed when the certificate is updated.
postgres-operator	<code>kubectl rollout restart deployment postgres-operator -n <i>&lt;namespace&gt;</i></code>
management-random-postgres*	Rollout restart all the postgres deployments: <code>kubectl -n <i>&lt;namespace&gt;</i> get deploy -o name   grep postgres   egrep -v "backrest pgbouncer operator"   xargs kubectl -n <i>&lt;namespace&gt;</i> rollout restart</code>
management-random-postgres-pgbouncer	<code>kubectl rollout restart deployment/<i>deployment_name</i> -n <i>&lt;namespace&gt;</i></code> A manual restart is not needed when the certificate is updated.
management-portal-proxy	<code>kubectl rollout restart deployment/<i>deployment_name</i> -n <i>&lt;namespace&gt;</i></code> A manual restart is not needed when the certificate is updated.
management-taskmanager	<code>kubectl rollout restart deployment/<i>deployment_name</i> -n <i>&lt;namespace&gt;</i></code> A manual restart is not needed when the certificate is updated.
management-tunnel	<code>kubectl rollout restart deployment/<i>deployment_name</i> -n <i>&lt;namespace&gt;</i></code>
management-turnstile	<code>kubectl rollout restart deployment/<i>deployment_name</i> -n <i>&lt;namespace&gt;</i></code>
management-ui	<code>kubectl rollout restart deployment/<i>deployment_name</i> -n <i>&lt;namespace&gt;</i></code>
management-remote-sitename-postgres	<code>kubectl rollout restart deployment/<i>deployment_name</i> -n <i>&lt;namespace&gt;</i></code>

3. Verify that all Management pods are ready by running the following command:

```
kubectl -n namespace get po
```

where *namespace* is the namespace where the Management subsystem is installed.

The restart is complete when all pods are `Running` and `Ready`.

## Gateway subsystem: restarting pods

Restart Gateway subsystem pods and components in your API Connect deployment.

## About this task

When you update Gateway subsystem secrets and certificates, the pods restart automatically.

## Portal subsystem: restarting pods

Restart Portal subsystem pods in your API Connect deployment.

### Procedure

1. If you are restarting pods after updating the `portal-ca`, complete the following steps:

a. Delete all of the portal pods with the following command:

In the command, `namespace` is the namespace where the Portal subsystem is installed:

- For a one replica deployment profile:

```
kubectl delete pods portal-sitename-db-0 portal-sitename-www-0 -n namespace
```

- For a three replica deployment profile:

```
kubectl delete pods portal-sitename-db-0 portal-sitename-db-1 portal-sitename-db-2 portal-sitename-www-0 portal-sitename-www-1 portal-sitename-www-2 -n namespace
```

b. Reload the NGINX configuration by running the following command:

```
kubectl -n namespace exec -it portal-tunnel -- openresty -s reload
```

2. If you updated any other certificate, you can perform a rolling restart on the pods by completing the following steps:

a. Determine the statefulset names by running the following command:

```
kubectl -n namespace get statefulset | grep CR_name
```

Where:

- `namespace` is the namespace where the Portal subsystem is installed.
- `CR-name` is the name of the CR used for installing the Portal subsystem.

b. Restart pods by running the appropriate `kubectl` commands, which are shown in Table 1. Use the statefulset name that you obtained in step 2a.

Table 1. Portal pods and the commands to restart them

Pod name	Restart command
portal-local_sitename-db	<code>kubectl rollout restart statefulset/statefulset_name</code>
portal-local_sitename-www	<code>kubectl rollout restart statefulset/statefulset_name</code>
portal-tunnel	<code>kubectl rollout restart statefulset/statefulset_name</code>
portal-remote_sitename-db	<code>kubectl rollout restart statefulset/statefulset_name</code>
portal-remote_sitename-www	<code>kubectl rollout restart statefulset/statefulset_name</code>
portal-nginx Attention: Restart this pod last.	<code>kubectl rollout restart statefulset/statefulset_name</code>

3. Verify that all Portal pods are ready by running the following command:

```
kubectl -n namespace get po
```

Where `namespace` is the namespace where the Portal subsystem is installed.

The restart is complete when all pods are **Running** and **Ready**.

## Analytics subsystem: restarting endpoints and pods

Restart the pods that support the Analytics service and endpoints in your API Connect deployment.

- [Restarting the Analytics ingestion endpoint](#)  
Restart the pods that support the Analytics ingestion endpoint in your API Connect deployment.
- [Restarting Analytics pods and data collection](#)  
Restart the pods that support the internal communications within the Analytics subsystem in your API Connect deployment.

## Restarting the Analytics ingestion endpoint

Restart the pods that support the Analytics ingestion endpoint in your API Connect deployment.

## About this task

---

The ingestion endpoint contains the external certificates that secure communications with the Analytics subsystem. The default names of these of the secrets for these certificates are `ingress-ca`, `analytics-ai-endpoint` (server), and `analytics-ingestion-client` (client).

When you refresh the certificates for the ingestion endpoint, restart the Analytics pods by completing the following steps.

## Procedure

---

1. **OVA users:** log in to your analytics VM and ssh to the root user:

```
ssh apicadm@<vm hostname>
sudo -i
```

2. Reload the NGINX configuration in the `analytics-mtls-gw` pod by running the following command:

```
kubectl -n namespace exec analytics-mtls-gw -- openresty -s reload
```

where:

- `namespace` is the namespace where Analytics is installed.
- `analytics-mtls-gw` is the name of the `analytics-mtls-gw` pod.

The Management and Gateway subsystems pick up the change automatically and do not require a restart.

3. Verify that the Analytics service is working by completing the following steps:
  - a. Invoke an API on your gateway that uses the associated Analytics subsystem.
  - b. Use the Analytics user interface to confirm that new data was posted to the Analytics service.

---

## Restarting Analytics pods and data collection

Restart the pods that support the internal communications within the Analytics subsystem in your API Connect deployment.

## Before you begin

---

This operation requires the use of the Analytics CLI. [Using the Analytics CLI](#).

## About this task

---

The internal certificates are used for all Analytics microservices to communicate with each other within the subsystem. The default names of the secrets for these are `analytics-ingestion-client`, `analytics-ai-endpoint`, `analytics-server`, and `analytics-ca`.

When you refresh the internal certificates for the Analytics subsystem, restart the Analytics pods by completing the following steps.

## Procedure

---

1. **OVA users:** log in to your analytics VM and ssh to the root user:

```
ssh apicadm@<vm hostname>
sudo -i
```

2. Restart all of the Analytics pods by running the following command:

```
kubectl -n namespace delete po -l app.kubernetes.io/instance=CR_name
```

where:

- `namespace` is the namespace where the Analytics subsystem is installed.
- `CR_name` is the name of the CR for installing the Analytics subsystem (default name: `analytics`).

3. Verify that all Analytics pods are ready by running the following command:

```
kubectl -n namespace get po
```

where `namespace` is the namespace where the Analytics subsystem is installed.

Make sure that every pod is **Running** and **Ready**.

4. Associate the Analytics service with the Gateway service.  
Note: If you did not unassociate the Analytics service from the Gateway, skip this step.  
See [Associating an analytics service with a gateway service](#).
5. Check the storage cluster's health again and wait until the status is **green**.
6. Verify that the Analytics service is working by completing the following steps:
  - a. Invoke an API on your gateway that uses the associated Analytics subsystem.
  - b. Use the Analytics user interface to confirm that new data was posted to the Analytics service.

---

## Re-enabling scheduled backups

If you disabled scheduled backups before updating keys, secrets, or passwords, enable the backups again for your API Connect deployment.

## About this task

---

### Procedure

---

1. Open the management cluster CR for editing by running the following command:

```
kubect1 -n namespace edit CR-name
```

where:

- **namespace** is the namespace where the Management subsystem is installed.
  - **CR-name** is the name of the CR that was used to install the Management subsystem.
2. In the **databaseBackup** section of the CR, replace the **schedule** entry that you removed earlier when you disabled scheduled backups.
  3. Save and close the file.

---

## The API Connect operations tool: apicops

Use the apicops operations tool to check the health, troubleshoot, and run maintenance operations on your API Connect installation.

The API Connect apicops tool is in active development and is available to download from [github.com](https://github.com). It is available for Linux® and macOS. All suggestions, feedback, and bug reports can be raised in this repository.

---

### Installing

Download the binary for your operating system from [latest release](#). Rename the file to **apicops**, and give it execute permission: **chmod +x apicops**.

---

### Running apicops

Apicops requires access to your Kubernetes or OpenShift environment. For more information about using apicops, see: [Requirements for running apicops](#).

---

## Configuring and managing your server environment

You configure and manage the servers that comprise your IBM® API Connect on-premises cloud by using the Cloud Manager user interface.

The Cloud Manager user interface is the part of IBM API Connect that enables a Cloud Administrator to define and manage the API Connect on-premises cloud.

You can use the Cloud Manager to define the API Connect cloud by performing the following tasks:

- Create Provider organizations and invite users to serve as the owner
- Create and manage user roles and role defaults
- Create availability zones for services
- Register the relevant servers that will provide the gateway, analytics, and portal services
- Associate an analytics service with a gateway to enable reports for API Events
- Configure resources for user authentication, TLS security, and OAuth providers and make the resources visible to all or selected provider organizations
- Connect to an existing SMTP mail server and edit templates for system-generated emails
- Set the default gateway service for catalogs
- [Cloud Manager configuration checklist](#)  
A summary, with links, of the key initial configuration tasks that you must complete in the Cloud Manager user interface after installing and deploying IBM API Connect, and further supplementary tasks.
- [Activating your Cloud Manager user account](#)  
If you have been invited to be an administrator of your API Connect cloud, you must activate your account by using the activation link that was sent by another administrator. You can then access the Cloud Manager user interface.
- [Accessing the Cloud Manager user interface](#)  
How to navigate to and login to the Cloud Manager user interface.
- [Defining your Cloud Manager topology](#)  
To define your API Connect on-premises cloud, you define availability zones and register services within those zones to securely create, promote, and track APIs. A Default Availability Zone is provided on installation.
- [Managing authentication and security](#)  
Secure Cloud Manager and API Manager as well as your Catalogs with a user registry. Secure your APIs with OAuth. Create TLS profiles to ensure that information you share among web servers will not be stolen or tampered with.
- [Configuring the cloud settings](#)  
Before you can add a provider organization to your cloud, you must specify your cloud settings to configure an email server for notifications as well as user registry options and catalog defaults.
- [Administering provider organizations](#)  
Manage the provider organizations who have accounts to publish APIs in your API Connect cloud.
- [Administering members and roles](#)  
Cloud Administrators can add members and assign them roles to enable them to work in Cloud Manager. The Cloud and Topology Administrators can also create roles and role defaults. You can also delete users to prevent them from accessing Cloud Manager.
- [Monitoring the cloud](#)  
API Connect generates events to monitor the status of your cloud.
- [Changing your Cloud Manager password and profile information](#)  
You can change your Cloud Manager password and update your profile information.

- [Resolving login problems by increasing HTTP header size](#)  
You can resolve login problems for the Cloud Manager UI by increasing the maximum HTTP client header size.
- [Onboarding a new admin for Cloud Pak for Integration](#)  
Use the API Connect toolkit CLI to create a new admin account for use with Cloud Pak for Integration.
- [Extending the Gateway server behavior](#)  
To support your enterprise requirements, you can extend the Gateway servers within IBM API Connect to provide extra enforcement behavior.
- [Cloud Manager Tutorials](#)  
Tutorials for using the Cloud Manager user interface in IBM API Connect.

## Cloud Manager configuration checklist

A summary, with links, of the key initial configuration tasks that you must complete in the Cloud Manager user interface after installing and deploying IBM® API Connect, and further supplementary tasks.

### Key initial tasks

Task	Description
<a href="#">Log in to the Cloud Manager user interface</a>	A default admin user account is provided for you to begin your configuration tasks.
<a href="#">Configure an email server for notifications</a>	The email server sends registration invitations and other event-driven emails. You must configure the email server before you add any provider organizations or register a Portal service.
<a href="#">Specify your email server as the notification server</a>	Select the email server to be used as the notification server, and configure the sender name and email address to be included in the emails.
<a href="#">Register a Gateway service</a>	A Gateway service represents a cluster of gateway servers that host published APIs and provide the API endpoints used by client applications. Gateways execute API proxy invocations to backend systems and enforce API policies including client identification, security and rate limiting.
<a href="#">Register an Analytics service</a>	The Analytics service collects API event data from the Gateway service.
<a href="#">Associate the Analytics service with a Gateway service</a>	You must associate an Analytics service with each of the Gateway services from which you want to collect API event data.
<a href="#">Register a Portal service</a>	The Portal service provides a developer portal used by application developers to discover APIs and onboard consumers.
<a href="#">Create a provider organization</a>	For developers to be able to publish APIs, and for API managers to be able to manage the API lifecycle, they must be a member of a provider organization. You must create a least one provider organization specifying the owner, who can add further members.

### Further supplementary tasks

Task	Description
<a href="#">Configure authentication and security</a>	API Connect provides default local user registries, but you can configure your own user registries, of various types, for user interface and API access control. You can also configure TLS profiles for securing data transmission over HTTPS, and OAuth providers for securing third-party website or application access to APIs.
<a href="#">Create an Availability Zone</a>	An Availability Zone is a logical grouping of one or more API Connect services. A default Availability Zone is provided on installation, but you can add further Availability Zones to reflect your data center topology.
<a href="#">Administer members and roles</a>	You can add further users as members of the administration organization, and control their permissions by assigning predefined roles. You can also create your own custom roles.
<a href="#">Monitor your API Connect cloud</a>	API Connect generates events to allow you to monitor the status of your cloud.

### Related concepts

- [Cloud Manager Tutorials](#)

## Activating your Cloud Manager user account

If you have been invited to be an administrator of your API Connect cloud, you must activate your account by using the activation link that was sent by another administrator. You can then access the Cloud Manager user interface.

### Before you begin

A cloud administrator invited you to join API Connect .

### Procedure

1. Complete the following steps to activate your Cloud Manager account:

If the Identity Provider uses LDAP

An invitation email with an activation link is sent. Click the activation link, or paste it in a browser, to log in directly with your LDAP user credentials. Upon authentication, the API Connect user record is updated from the backend identity provider.

If the Identity Provider uses a local registry

An invitation email with an activation link is sent. Click the activation link or paste it in a browser. The activation link takes you to a sign up page where you enter your first name, last name and password. Passwords must have a minimum of 8 characters and contain characters from at least three of the four following categories:

- Uppercase letters
- Lowercase letters
- Numbers
- Special characters (for example: ! # \$ %)

Note:

- The email address that you enter on the sign up page must match the email address to which the invitation email was sent, otherwise the account activation fails.
- If you previously had an account that was removed, and you are being invited again, you must re-activate your account by using the Sign In option on the page, **not** by completing the registration form and using the Sign Up option; attempting to re-register will fail.

If the Identity Provider uses an authentication URL

An invitation email with an activation link is sent. Click the activation link or paste it in a browser. The activation link takes you to a sign up page where you enter your credentials, which then become your user name and password. Upon authentication, the API Connect user record is updated from the backend identity provider.

If multiple user registries are available for selection on the sign up page, then make sure that the correct registry for your Cloud Manager account is selected. You might need to ask your administrator which user registry is appropriate for your account.

For information on configuring user registries and making them available for Cloud Manager login, see [User registries overview](#) and [Selecting user registries for Cloud Manager and API Manager](#).

2. Click Sign up to complete your registration, then click Sign in to open the Cloud Manager login page.

## Results

---

You have created your API Connect administrator account.

## What to do next

---

Log in to the Cloud Manager user interface and, depending on your role, start to manage the cloud topology, and work with provider organizations and cloud resources. To access the Cloud Manager login page in the future, use the following URL:

```
https://host/admin
```

where *host* is the fully qualified host name or IP address of the Management server.

## Related concepts

---

- [Managing authentication and security](#)
- [Configuring the cloud settings](#)

## Related tasks

---

- [Accessing the Cloud Manager user interface](#)
- [Defining your Cloud Manager topology](#)
- [Administering provider organizations](#)
- [Administering members and roles](#)

# Accessing the Cloud Manager user interface

---

How to navigate to and login to the Cloud Manager user interface.

## Procedure

---

1. Open a browser and enter the URL for the cloud-admin-ui endpoint followed by /admin. Note that this is a secure connection that uses HTTPS, for example:

```
https://<cloud-admin-ui>/admin.
```

- VMware installations: The cloud-admin-ui endpoint was set during installation of the Management subsystem by the following command: `apicup subsys set mgmt cloud-admin-ui <endpoint>`. Append /admin to this value.
- Kubernetes, OpenShift, Cloud Pak for Integration installations: The cloud-admin-ui endpoint is specified in the Management subsystem CR, in the `Endpoints` section, for example:

```
Endpoints:
...
Name: admin
Type: UI
Uri: https://example.com/admin
...
```

2. Enter the Cloud Administrator user name and password.

- Kubernetes, VMware, and OpenShift individual Management subsystem install:  
If you are logging in for the first time, enter `admin` for the user name and `7iron-hide` for the password. Otherwise, enter `admin` for the user name and your own password.



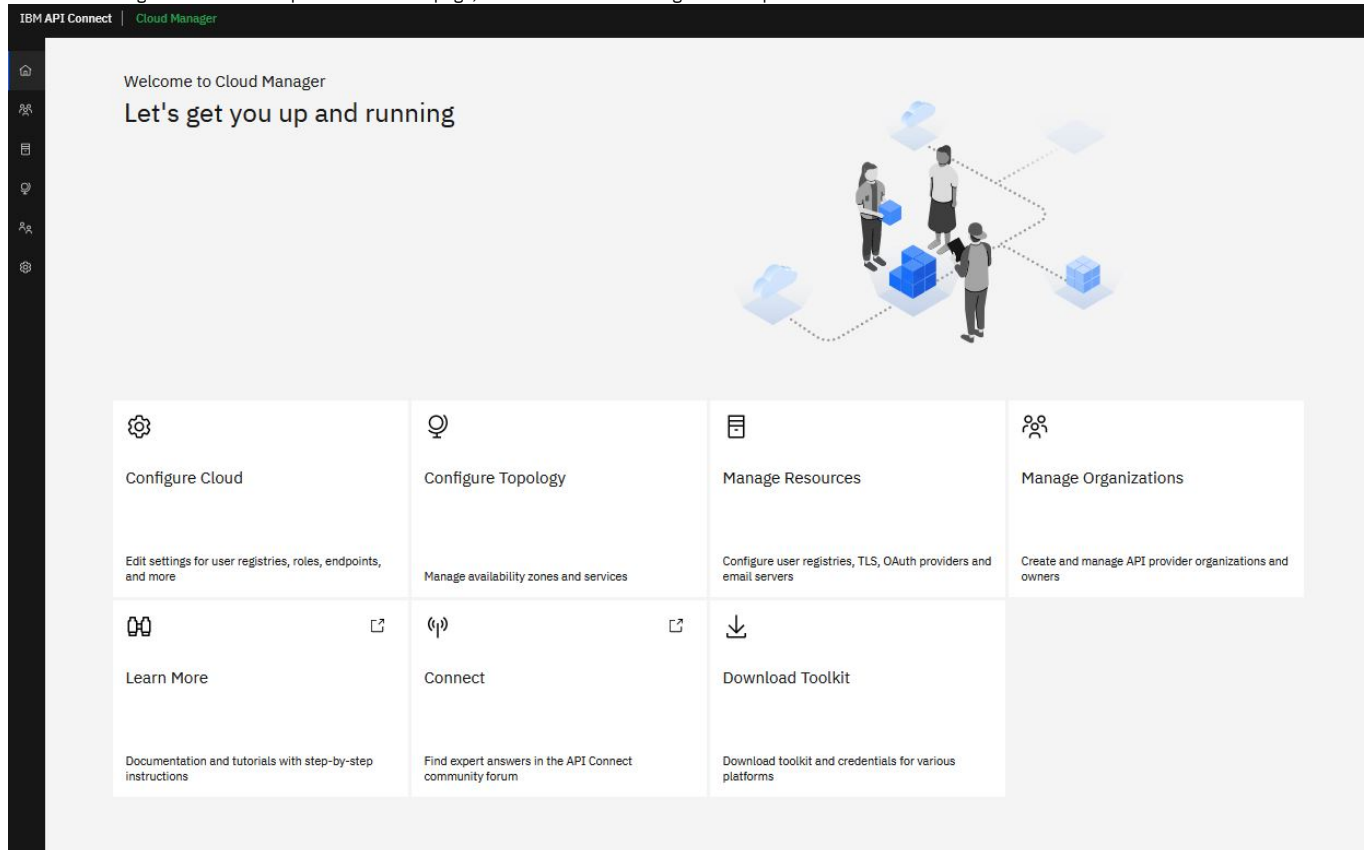
- OpenShift top-level CR install:  
The default user name is `admin`. If you are logging in for the first time, run the following commands to retrieve the initial password, which is generated automatically during deployment:

```
oc get secret -n <namespace> | grep mgmt-admin-pass
oc get secret -n <namespace> <secret_name_from_previous_command> -o jsonpath="{.data.password}" | base64 -d && echo
""
```

3. If your cloud contains multiple user registries, as configured on the Settings > User Registries screen, then choose the user registry that contains your login credentials. You might need to ask your administrator which user registry is appropriate for your account.
4. Click Sign In.  
Important: The first time that you access the cloud console, you must, for security reasons, enter a new password and your email address. If you forget your password and request a password reset, the notification email is sent to this email address. The email is sent by the email server that is configured in the **Notifications** section of the cloud **Settings**.

## Results

The Cloud Manager user interface opens to the home page, as shown in the following screen capture:



## What to do next

- Configure the email server so you can reset your password if needed and send other notifications. See [Tutorial: Configuring the Cloud](#).
- Configure your topology by registering services. See [Tutorial: Configuring the Cloud](#).
- Create a Provider Organization. See [Tutorial: Creating a Provider Organization](#).

## Related tasks

- [Changing your Cloud Manager password and profile information](#)

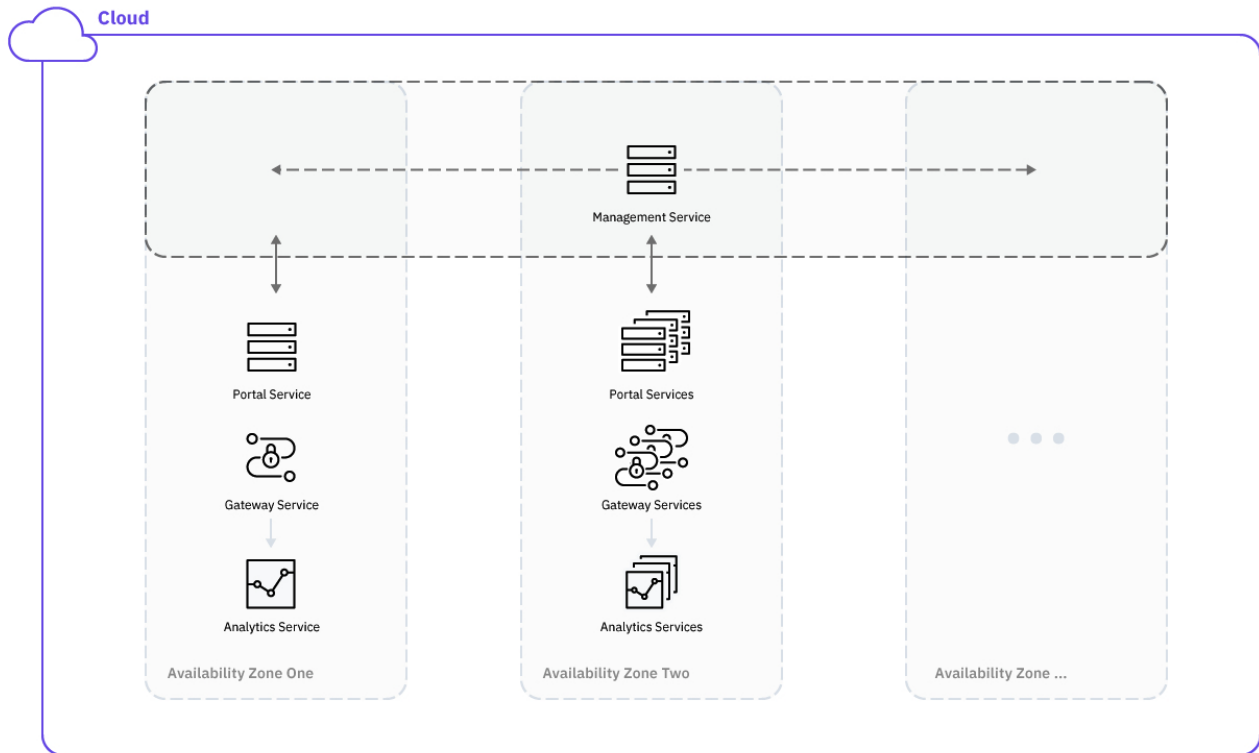
## Defining your Cloud Manager topology

To define your API Connect on-premises cloud, you define availability zones and register services within those zones to securely create, promote, and track APIs. A Default Availability Zone is provided on installation.

## About this task

The Cloud Manager topology consists of Availability Zones that contain the API Connect services. Availability Zones can contain one or more gateway services, analytics services, and portal services.

The following diagram illustrates the topology for services and Availability Zones in an API Connect network:



One of the following roles is required to register and manage services:

- Administrator
- Topology Administrator
- Owner
- A custom role with the **Topology:Manage** permission

Following are the tasks involved in defining the API Connect topology:

- Create one or more Availability Zones.
- Register one or more Gateway, Analytics, and Portal services in each Availability Zone
- Associate an Analytics service with each Gateway service
- Set the visibility for the services

Once the cloud is defined and an email server has been configured, you invite other users to access API Manager to create APIs. You expose these APIs to the development community through the Developer Portal user interface.

To define the API Connect cloud, complete the following tasks:

- **Creating an availability zone**  
An Availability Zone is a logical group of one or more API Connect services. The Default Availability Zone is created during installation.
- **Registering a gateway service**  
A gateway service is required to handle incoming traffic for APIs.
- **Registering an analytics service**  
Configure at least one analytics service in your API Connect on-premises cloud. The analytics service is always associated with a gateway service, from which it collects API event data.
- **Registering a portal service**  
Define one or more Portal services in your API Connect on-premises cloud.
- **Associating an analytics service with a gateway service**  
To collect data about your API usage and other statistics, you must associate an analytics service with each gateway service. You can associate multiple gateway services with one analytics service, but each gateway can be associated with only one analytics service.
- **Setting visibility for a service**  
The visibility setting determines which provider organizations can access a service.

## Related concepts

- [Configuring and managing your server environment](#)

## Creating an availability zone

An Availability Zone is a logical group of one or more API Connect services. The Default Availability Zone is created during installation.

## About this task

Availability zones allow you to logically group API Connect services to suit your environment. For example, you can group gateway services according to the region or data center they are located in.

The Default Availability Zone is created during the installation process.

When you create an availability zone, you should provide a descriptive title; a name is generated automatically for internal identification. For example:

Table 1. Example availability zones


Title	Name
US Availability Zone	us-availability-zone
Hampshire Availability Zone	hampshire-availability-zone
Newport Availability Zone	newport-availability-zone
London South Availability Zone	london-south-availability-zone

One of the following roles is required to add and manage Availability Zones:

- Administrator
- Topology Administrator
- Owner
- A custom role with the **Topology:Manage** permission

## Procedure

Follow these steps to add one or more additional Availability Zones to your on-premises cloud:

1. In the Cloud Manager, click  Topology.
2. You will see the current Availability Zones configured in your cloud. The Default Availability Zone is created at installation. To add another Availability Zone, choose Create Availability Zone.
3. Enter the following values:

Field	Description
Title (required)	Enter a descriptive title for the availability zone. This title will display on the screen.
Name (required)	This field is auto-populated by the system and used as the internal field name.
Summary (optional)	Enter a brief description.

4. Click Create to complete the operation.

## Results

The availability zone will be added on the Topology page. You can now add gateway, analytics, and portal services to the availability zone.

## Related tasks

- [Registering a gateway service](#)
- [Registering an analytics service](#)
- [Registering a portal service](#)

## Registering a gateway service

A gateway service is required to handle incoming traffic for APIs.

## Before you begin

Before registering a Gateway service in Cloud Manager, the DataPower® API Connect Gateway Service has to either be installed as a subsystem in your Kubernetes cluster or enabled on the DataPower appliance. For a Kubernetes environment, see [DataPower Gateway subsystem on Kubernetes](#). For appliances, see [Configuring the API Connect Gateway Service](#) in the appropriate version of the [DataPower documentation](#).

Note:

If you enabled Event Endpoint Management in Cloud Pak for Integration, an Event Gateway Service is provided for event endpoints. For more information about registering an Event Gateway Service, see the [Cloud Pak for Integration documentation](#).

## About this task

A Gateway service represents a cluster of gateway servers that host published APIs and provide the API endpoints used by client applications. Gateways execute API proxy invocations to backend systems and enforce API policies including client identification, security and rate limiting.


One of the following roles is required to register and manage a gateway services:


- Administrator
- Topology Administrator
- Owner
- A custom role with the **Topology:Manage** permission

Note: You can also register, and manage, gateway services by using the developer toolkit CLI; for details, see [apic\\_gateway-services](#).

## Procedure

Complete the following steps to configure a Gateway service for your cloud:

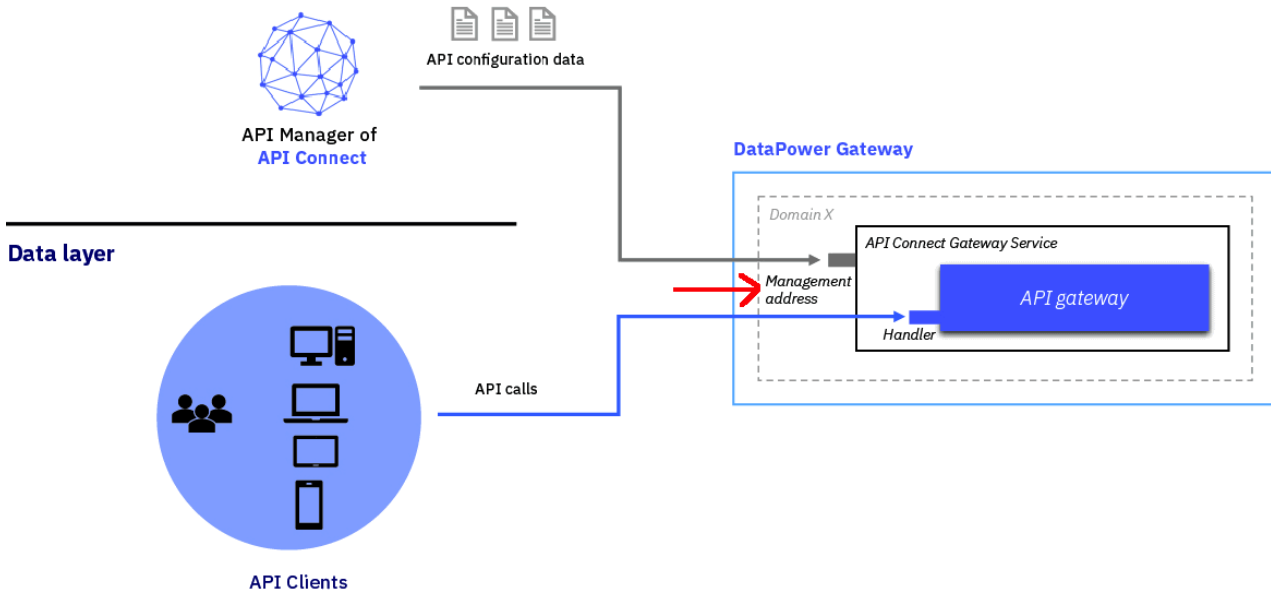
1. In the Cloud Manager, click  Topology.
2. From the Availability Zone that will contain the Gateway service, select Register Service.
3. On the Configure Service page, select DataPower Gateway as the service type. Select the Gateway type that you want to create, either DataPower Gateway (v5 compatible) or DataPower API Gateway. For a description of the gateway types, see [API Connect gateway types](#)
4. Enter the values to configure the Gateway service. You will need to obtain the endpoints from your deployment configuration. For a Kubernetes environment, the endpoints are configured by the following values in the CR (custom resource) file. For an appliance, the endpoint is configured in DataPower.

Field	Description
Title (required)	Enter a descriptive title for the gateway service. This title will be displayed on the screen.
Name (required)	This field is auto-populated by the system and used as the internal field name.
Summary (optional)	Enter a brief description.
Management endpoint on the gateway service: Endpoint (required)	<p>Enter the API Connect Gateway Service endpoint.</p> <ul style="list-style-type: none"> <li>• For a Kubernetes environment, the Management Endpoint is the endpoint created in the CR (custom resource). See <a href="#">DataPower Gateway subsystem on Kubernetes</a> for more information.</li> <li>• For an appliance, the Management Endpoint is the <i>Management address</i> to the API Connect Gateway Service shown in the <a href="#">gateway service connection diagram</a>. For one gateway, this takes the form <code>http://&lt;ip-address-for-gateway&gt;:3000</code>. For multiple gateways, it would be the address:port of the load balancer</li> <li>• If you are using <code>in-cluster</code> communication, set this to the service endpoint URL for the gateway, which has the format:           <pre>https://&lt;name&gt;.&lt;namespace&gt;.svc</pre> <p>where:</p> <ul style="list-style-type: none"> <li>◦ <code>&lt;name&gt;</code> is the <code>name</code> property from the gateway subsystem CR. If you are using the top-level CR this is <code>&lt;apic instance name&gt;-gw</code></li> <li>◦ <code>&lt;namespace&gt;</code> is the namespace that the gateway CR exists in.</li> </ul> </li> </ul>
Management endpoint on the gateway service: TLS client profile (required)	<p>Specify the TLS Client profile to use when contacting the gateway through the management endpoint.</p> <p>Note:</p> <ul style="list-style-type: none"> <li>• The default value is Gateway management client TLS client profile:1.0.0. If you are using a DataPower Gateway in a Kubernetes deployment, with mutual TLS enabled, use this setting. For any other gateway, if the gateway is not configured with the correct TLS credentials the gateway registration will fail with this setting. To prevent registration failure, select either Default TLS client profile:1.0.0 or your own custom TLS client profile. For information on configuring a custom TLS client profile, see <a href="#">Creating a TLS Client Profile</a>.</li> <li>• TLS protocol version 1.3 is not supported.</li> </ul>
Management endpoint on the gateway service: Use <code>in-cluster</code> communication switch	If you want to use <code>in-cluster</code> communication between the management and gateway subsystems, then enable this switch. For more information about this option, see <a href="#">In-cluster or external communication between subsystems</a> . If you are not sure, then leave this switch disabled (the default).
Management endpoint on the gateway service: Use JWT for gateway authentication to analytics service. Available in v10.0.5.4 and later.	If you plan to associate an analytics service with this gateway, and you want to use JWT authentication between the gateway and analytics service, then enable this switch. If this switch is not enabled, then mutual TLS authentication is used between the gateway and analytics service. For more information about JWT security, see <a href="#">Enable JWT security instead of mTLS</a>
<b>API invocation endpoint:</b> API endpoint base (required)	<p>Enter the base portion of the URL that maps to the base portion of the URL for incoming API traffic. It is a public FQDN with additional paths that are specific to your API calls. For example: <code>https://api.mycompany.com</code></p> <p>Note: If you are using the DataPower API Gateway, HTTP/2 is enabled automatically on the API invocation endpoint. If you are using the DataPower Gateway (v5 compatible), HTTP/2 is not supported on the API invocation endpoint.</p>
<b>API invocation endpoint:</b> Server Name Indication (SNI) - Host name	<p>For supporting Server Name Indication (SNI) at the API Endpoint Base. The default hostname of <code>***</code> is required to allow all hosts. Enter other host names as needed. Wild card format is supported. The SNI capability enables you to serve multiple TLS secure host names through the same Gateway service, using the same IP address and port, without requiring them to use the same TLS profile.</p> <p>Note: To allow requests from clients that don't support SNI, you <b>must</b> include a host name value of <code>***</code>.</p>
<b>API invocation endpoint:</b> Server Name Indication (SNI) - TLS server profile	<p>The TLS server profile that supports the given hostname for SNI.</p> <p> Restriction: If you are using the DataPower API Gateway, HTTP/2 is enabled automatically on the API invocation endpoint. Therefore, using a TLS server profile that supports only TLS protocol version 1.3 is not supported and will result in the gateway being unable to receive API traffic. You must use a TLS server profile that includes TLS protocol version 1.2 as one of its supported protocols.</p> <p>This restriction does not apply if you are using the DataPower Gateway (v5 compatible), which does not support HTTP/2.</p>
OAuth shared secret (optional)	For sites using native OAuth providers, enter the shared secret that will be used by all API calls going through the gateway.
	Note: The specified shared secret must be 64 characters (64 bytes) in length, prefixed with <code>0x</code> , and must consist only of hexadecimal characters. For example: <code>0xa354282f227c10250511ae9c9e8c7ed9f4f1bd0d7c04cb6d5bd178f8c62296e3</code>

The following diagram illustrates the gateway service connection:

---

## Management layer



5. When you are finished, click Save.

---

## Results

The Gateway service is added to the appropriate Availability Zone for your cloud.

---

## What to do next

Add additional gateway services. Add one or more analytics services. Add one or more portal services. Associate the gateway service with an analytics service. Set the visibility for the gateway.

---

## Related tasks

- [Registering a portal service](#)
- [Registering an analytics service](#)
- [Associating an analytics service with a gateway service](#)
- [Setting visibility for a service](#)

---

## Registering an analytics service

Configure at least one analytics service in your API Connect on-premises cloud. The analytics service is always associated with a gateway service, from which it collects API event data.

---

## Before you begin

You must complete the following tasks:

- [Creating an availability zone](#)
- [Registering a gateway service](#)

---

## About this task

The analytics service collects API event data from the gateway service. In Cloud Manager, you configure the analytics service and then associate it with one or more gateways. Multiple gateways can be associated with a single analytics service, but each gateway can be associated with only one analytics service.

One of the following roles is required to register and manage analytics services:

- Administrator
- Topology Administrator
- Owner
- A custom role with the **Topology:Manage** permission

When you register the analytics service, you must select from one of three inter-subsystem communication methods. Before you decide which method to use, familiarize yourself with the difference between **external** and **in-cluster** communication: [In-cluster or external communication between subsystems](#). If you are an OVA user, or are in doubt as to what communication method to use, then use the default **external** setting.

The analytics subsystem has two separate communication flows:

- Management to analytics. The management service communicates with the analytics subsystem to query for analytics data to present in the UI dashboards. This flow is known as "analytics queries".
- Gateway to analytics. The gateway sends API event data to the analytics service. This flow is known as "analytics ingestion".

What you select depends on what is most suitable for your network configuration. The options are:

Use **external** communication for both ingestion and queries

This option is the default, and is what is used for analytics services that are upgraded from a pre-10.0.5.3 release. For OVA deployments, or if your analytics subsystem is in a different environment then **external** is the only option that you can use. Both the gateway and the management service communicate with the analytics service by using the external endpoint, which is a Kubernetes **ingress** or OpenShift **route** depending on your platform.

Use **in-cluster** communication for both ingestion and queries


You can select **in-cluster** if you have a Kubernetes or OpenShift deployment where all subsystems are within the same cluster. When this option is selected, the management, gateway, and analytics subsystems communicate with each other through internal service endpoints rather than externally accessible **ingresses** (Kubernetes) or **routes** (OpenShift).

Use **in-cluster** for ingestion and **external** for queries, or vice versa

You can select this option if you have a Kubernetes or OpenShift deployment and you have some of your subsystems installed in the same cluster. When this option is selected, you can use different communication methods for the different analytics communication flows. For example, the management to analytics communication can use **in-cluster**, and the gateway to analytics can use **external**. You might choose this configuration if your gateway is in a different cluster to the rest of your subsystems.

## Procedure

Complete the following steps to configure the analytics services for your cloud:

1. In the Cloud Manager, click  Topology.
2. Select Register Services > Analytics in the Availability Zone where you have the gateways you want to associate with the analytics service.
3. Enter the values to configure the analytics service:

Field	Description																								
Title	Enter a descriptive title for the analytics service.																								
Name	This field is auto-populated by the system and used as the internal field name.																								
Summary	Enter a brief description.																								
Service endpoint configuration: Communication method.	Select your communication method, for more information on the options see, <a href="#">Communication types</a> . If your API Connect deployment is OVA then you must select the first option: Use <b>external</b> communication for both ingestion and queries.																								
Service endpoint configuration: <b>External</b> analytics endpoint.	<p>Enter the fully qualified domain name for the analytics ingestion endpoint that you defined during installation. To determine the endpoint, run the following command:</p> <ul style="list-style-type: none"> <li>• Kubernetes           <pre>kubectl get ingress -n &lt;namespace&gt;</pre> <p>The endpoint has <b>-ai-</b> in its name:</p> <table border="1"> <thead> <tr> <th>NAME</th> <th>CLASS</th> <th>HOSTS</th> <th>ADDRESS</th> <th>PORTS</th> <th>AGE</th> </tr> </thead> <tbody> <tr> <td>def-analytics-ai-endpoint</td> <td>nginx</td> <td>a7s-in.example.com</td> <td></td> <td>80, 443</td> <td>87d</td> </tr> </tbody> </table> </li> <li>• OpenShift           <pre>oc get routes -n &lt;namespace&gt;</pre> <p>The endpoint has <b>-ai-</b> in its name:</p> <table border="1"> <thead> <tr> <th>NAME</th> <th>CLASS</th> <th>HOSTS</th> <th>ADDRESS</th> <th>PORTS</th> <th>AGE</th> </tr> </thead> <tbody> <tr> <td>def-analytics-ai-endpoint</td> <td>nginx</td> <td>a7s-in.example.com</td> <td></td> <td>80, 443</td> <td>87d</td> </tr> </tbody> </table> </li> <li>• VMware           <pre>apicup subsys get &lt;analytics subsystem&gt;</pre> <p>The endpoint is the value of property <b>analytics-ingestion</b>.</p> </li> </ul> <p>This field is disabled if "Use <b>in-cluster</b> communication for both ingestion and queries" is set as the communication type.</p>	NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE	def-analytics-ai-endpoint	nginx	a7s-in.example.com		80, 443	87d	NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE	def-analytics-ai-endpoint	nginx	a7s-in.example.com		80, 443	87d
NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE																				
def-analytics-ai-endpoint	nginx	a7s-in.example.com		80, 443	87d																				
NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE																				
def-analytics-ai-endpoint	nginx	a7s-in.example.com		80, 443	87d																				
Service endpoint configuration: TLS client profile for <b>external</b> analytics endpoint.	Select the TLS Client Profile to use for management to analytics subsystem communication. The Client Profile applies to the endpoint that you specified in the previous row. The default is "Analytics Ingestion TLS client profile". This field is disabled if "Use <b>in-cluster</b> communication for both ingestion and queries" is set as the communication type.																								
Service endpoint configuration: <b>In-cluster</b> analytics endpoint.	<p>Enter the fully qualified domain name for the analytics ingestion service endpoint. The format of the service endpoint is</p> <pre>https://&lt;name&gt;.&lt;namespace&gt;.svc</pre> <p>where:</p> <ul style="list-style-type: none"> <li>• <b>&lt;name&gt;</b> is the <b>name</b> property from the analytics subsystem CR. If you are using the top-level CR this is <b>&lt;apic instance name&gt;-a7s</b>.</li> <li>• <b>&lt;namespace&gt;</b> is the namespace that the analytics CR exists in.</li> </ul> <p>This field is disabled if Use <b>external</b> communication for both ingestion and queries is set as the communication type.</p>																								

Field	Description
Service endpoint configuration: TLS client profile for <b>in-cluster</b> endpoint.	Select the TLS Client Profile to use for management to analytics subsystem communication. The Client Profile applies to the Endpoint that you specified in the previous row. The default is "Analytics Ingestion TLS client profile". This field is disabled if Use <b>external</b> communication for both ingestion and queries is set as the communication type.
Service endpoint configuration: Use <b>in-cluster</b> endpoint for analytics queries.	If you want the management subsystem to use the <b>in-cluster</b> service endpoint to communicate with the analytics subsystem, then select this option. If this option is not selected, then the management subsystem uses the <b>external</b> endpoint. Selection of <b>external</b> or <b>in-cluster</b> communication between the gateway and analytics subsystem is done when the gateway is associated with the analytics service, see <a href="#">Associating an analytics service with a gateway service</a> . This setting is enabled when "Use <b>in-cluster</b> for ingestion and <b>external</b> for queries, or vice versa" is selected.

4. Click Save to complete the operation.

## Results

The analytics service is added to your cloud settings.

## What to do next

- Set preferred retention and rollovers settings for the new analytics service: [Configuring data retention and index rollover time periods](#).
- Associate the analytics service with one or more gateway services. For more information, see [Associating an analytics service with a gateway service](#).

## Related tasks

- [Registering a gateway service](#)
- [Registering a portal service](#)

## Registering a portal service

Define one or more Portal services in your API Connect on-premises cloud.

## Before you begin

You must complete the following tasks:

- [Creating an availability zone](#)
- [Registering a gateway service](#)
- [Configuring an email server for notifications](#)
- [Setting up notifications](#)

Note:

When you create or register a Developer Portal service, the Portal subsystem checks that the Portal web endpoint is accessible. However sometimes, for example due to the complexity of public and private networks, the endpoint cannot be reached. The following example shows the errors that you might see in the `portal-www` pod, admin container logs, if the endpoint cannot be reached:

```
An error occurred contacting the provided portal web endpoint: example.com
The provided Portal web endpoint example.com returned HTTP status code 504
```

In this instance, you can disable the Portal web endpoint check so that the Developer Portal service can be created successfully. To disable the endpoint check, complete one of the following updates depending on your platform:

On Kubernetes, OpenShift, and IBM® Cloud Pak for Integration  
Add the following section to the Portal custom resource (CR) template:

```
spec:
 template:
 - containers:
 - env:
 - name: PORTAL_SKIP_WEB_ENDPOINT_VALIDATION
 value: "true"
 name: admin
 name: www
```

On VMware

In your `apicup` project, create a file called `ptl-extra-values.yaml` (or edit the file if one already exists), and add the following section:

```
spec:
 template:
 - containers:
 - env:
 - name: PORTAL_SKIP_WEB_ENDPOINT_VALIDATION
 value: "true"
 name: admin
 name: www
```

Run the following commands:

```
apicup subsys set <ptl_subsys> extra-values-file <path-to-ptl-extra-values-yaml-file>
apicup subsys install <ptl_subsys>
```

## About this task


Each Availability Zone contains one or more Portal services. The Portal service provides a Developer Portal used by application developers to discover APIs and onboard consumers. An email server must be configured and set as the email server for the cloud before registering a Portal service. One of the following roles is required to register and manage Portal services:

- Administrator
- Topology Administrator
- Owner
- A custom role with the **Topology:Manage** permission

Important: It's not recommended to have more than 100 sites per Developer Portal service. Note that it's not necessary to have a Portal site for every Catalog, for example Catalogs that are only for API Developers don't need a Portal site, as the APIs can be tested by using credentials from the API Manager. If more than 100 sites are required, you should configure additional Developer Portal services.

## Procedure

Complete the following steps to configure the Portal services for your cloud:

1. In the Cloud Manager, click  Topology.
2. In the Availability Zone that will contain the Portal service, select Register Services > Portal.
3. Enter the values to configure the Portal service.

Field	Description
Title (required)	Enter a descriptive title for the portal service. This title will display on the screen.
Name (required)	This field is auto-populated by the system and used as the internal field name.
Summary (optional)	Enter a brief description.
Management endpoint on the portal service: Endpoint (required)	Enter the IP address, fully-qualified host name, service, or ingress name. Used for communication with API Manager. <ul style="list-style-type: none"><li>• In a Kubernetes deployment, the endpoint is defined in the CR by <code>api.portal.\$STACK_HOST</code>. See <a href="#">Developer Portal subsystem on Kubernetes</a>.</li><li>• If configured during installation in a VMware environment, it is the <code>apicup subsys set portal portal-admin</code> value.</li><li>• If you are using <code>in-cluster</code> communication, set this to the service endpoint URL for the portal, which has the format: <pre>https://&lt;name&gt;.&lt;namespace&gt;.svc</pre>where:<ul style="list-style-type: none"><li>◦ <code>&lt;name&gt;</code> is the <code>name</code> property from the portal subsystem CR. If you are using the top-level CR this is <code>&lt;apic instance name&gt;-ptl</code></li><li>◦ <code>&lt;namespace&gt;</code> is the namespace that the portal CR exists in.</li></ul></li></ul>
Management endpoint on the portal service: TLS client profile (optional)	Select the TLS Client Profile that will be used to communicate with the portal service. The profile applies to the Management Endpoint. Note: TLS protocol version 1.3 is not supported.
Management endpoint on the portal service: Use <code>in-cluster</code> communication switch	If you want to use <code>in-cluster</code> communication between the management and portal subsystems, then enable this switch. For more information about this option, see <a href="#">In-cluster or external communication between subsystems</a> . If you are not sure, then leave this switch disabled (the default).
Portal website URL (required)	The URL that will be used for public access to the portal. <ul style="list-style-type: none"><li>• In a Kubernetes deployment, the endpoint is defined in the CR by <code>portal.\$STACK_HOST</code>. See <a href="#">Developer Portal subsystem on Kubernetes</a>.</li><li>• If configured during installation in a VMware environment, it is the <code>apicup subsys set portal portal-www &lt;portal&gt;.&lt;hostname&gt;.&lt;domainname&gt;</code> value.</li></ul> Multiple <code>portal-www</code> endpoints may be configured, as described here in these topics: <a href="#">Defining multiple portal endpoints for a Kubernetes environment</a> and <a href="#">Defining multiple portal endpoints for a VMware environment</a> .

4. When you are finished, click Save.

## Results

The Portal service is configured in your cloud and can be used to publish APIs.

## Related tasks

- [Registering a gateway service](#)
- [Registering an analytics service](#)

## Associating an analytics service with a gateway service

To collect data about your API usage and other statistics, you must associate an analytics service with each gateway service. You can associate multiple gateway services with one analytics service, but each gateway can be associated with only one analytics service.

## About this task



The associated analytics service receives API event data from the gateway service.

To associate an analytics service with a gateway, you must first configure at least one analytics service and one gateway service.


One of the following roles is required to associate an analytics service with a gateway service:

- Administrator
- Topology Administrator
- Owner
- A custom role with the **Topology:Manage** permission

Follow these steps to associate an analytics service with a gateway service:

## Procedure

---

1. In the Cloud Manager, click  Topology.
2. In the section for the Availability Zone you want to work on, locate the gateway service that requires an analytics service in the SERVICE column.
3. Locate the ASSOCIATED ANALYTICS SERVICE column.
4. If an analytics service is configured, the Associate Analytics service link is enabled for each gateway service. Click Associate Analytics Service in the same row as the gateway service that requires an analytics service.
5. In Associate Analytics Service, you see the name of the gateway service and a list of analytics services that have been configured in your cloud.
6. Select the analytics service to associate with the gateway service by adding a checkmark next to the analytics services.
7. If you specified **in-cluster** communication when you registered the analytics service, you can enable the gateway to send API event data to the internal analytics endpoint. Enable the "Use internal endpoint for ingestion of analytics data from the gateway service" switch. For more information about **in-cluster** communication, see [Communication types](#).
8. Click Associate.
9. To remove the associated analytics service, select Unassociate analytics service from the actions menu. The gateway stops sending API event data to the analytics service.

## Results

---

The analytics service is associated with a gateway service and starts to receive API event data from that gateway.

## Related tasks

---

- [Registering a gateway service](#)
- [Registering an analytics service](#)

## Related information

---

- [Installing API Connect into a Kubernetes environment](#)
- [Deploying the Management OVA file](#)

## Setting visibility for a service

---

The visibility setting determines which provider organizations can access a service.

## Before you begin

---

You must complete the following tasks:

- Create additional availability zones if needed, see [Creating an availability zone](#)
- Register at least one gateway service, see [Registering a gateway service](#)
- Register other services.

## About this task

---

The visibility setting controls which provider organizations can use a service. The default visibility setting is Public.


One of the following roles is required to set the visibility for services:

- Administrator
- Topology Administrator
- Owner
- A custom role with the **Topology:Manage** permission

## Procedure

---

Follow these steps to set the visibility for the services in your on-premises cloud:

1. In the Cloud Manager, click  Topology.
2. From the list of Services, choose Set visibility from the actions menu next to the name of the service that requires the visibility setting.
3. Select the visibility setting for the service. The options are:
  - Private - the service is not visible and cannot be used by any provider organization

- Public - the service is visible and can be used by all provider organizations
  - Custom - the service is visible only to the provider organizations designated by you
4. For Custom visibility, select the provider organizations that will be able to use the service.
  5. Click Make visible to complete the operation.

## Results

---

For Private, the service cannot be used by any provider organizations. For Public, the service can be used by all provider organizations. For Custom, the service can be used by the provider organizations that you designate.

## Managing authentication and security

---

Secure Cloud Manager and API Manager as well as your Catalogs with a user registry. Secure your APIs with OAuth. Create TLS profiles to ensure that information you share among web servers will not be stolen or tampered with.

As Cloud Administrator or Topology Administrator (or with a custom role that contains the `Settings:Manage` permission), you can configure the following authentication and security mechanisms:

- User registries to authenticate users of Cloud Manager and API Manager and of your Catalogs and APIs.
- OAuth providers to provide protection for APIs.
- TLS profiles to secure transmission of data through the Gateway to external web sites and among web servers.

The following topics describe how to configure user registries, OAuth providers, and TLS profiles:

- [User registries overview](#)  
API Connect supports several types of user registries for authenticating users. The credentials for all users of Cloud Manager, API Manager, and the Developer Portal must be stored in a user registry.
- [TLS profiles overview](#)  
API Connect supports TLS Profiles for securing data transmission over HTTPS.
- [OAuth Provider overview](#)  
API Connect supports OAuth Specification 2.0, for both Native and Third party implementations.

## Related concepts

---

- [User registries overview](#)
- [OAuth Provider overview](#)
- [TLS profiles overview](#)

## User registries overview

---

API Connect supports several types of user registries for authenticating users. The credentials for all users of Cloud Manager, API Manager, and the Developer Portal must be stored in a user registry.

## Introduction to user registries

---

A user registry holds unique user account credentials, primarily usernames and passwords, which are accessed during authentication for logging into Cloud Manager, API Manager, the Developer Portal, and also for calling APIs (if the API is configured to use Basic Authentication in the Security Definition). The Cloud Administrator configures user registries to ensure that all users are successful at logging in.

User registries also provide security when calling APIs. For information on configuring Security Definitions for APIs, see [Creating a basic authentication security definition](#).

User registries are configured as Resources. Once the user registry has been configured as a resource in Resources > User registries, the next step is indicate the active user registries for your cloud in Settings > User registries. The user registry must be made active in your cloud in order to make it available for authentication. When a user logs into Cloud Manager or API Manager, the specified user registry is queried for credentials to confirm the user's identity. To make a user registry available to the Developer Portal, you must define the registry for consumer onboarding in the associated Catalog. In the API Manager UI, click Manage followed by the relevant Catalog, and then click Settings > Onboarding; for more information see [Creating and configuring Catalogs](#).

**Important:** Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access. In Cloud Manager and API Manager, a registry cannot be changed after a user is invited to be the owner of a provider organization, even if the invitation is not yet accepted.

User registries in API Connect serve the following primary functions:

- To authenticate a user at login time based upon username and password.
- To store basic profile information such as first name, last name, and email address.
- To provide secure access to Catalogs.
- To provide Basic Authentication for APIs when called by an application.

In order to log in to Cloud Manager, API Manager, the Developer Portal, or access a catalog, a user must have valid credentials (username and password) stored in a user registry that is configured in API Connect.

The Security Definition for an API can be configured to require a username and password, called the Basic Authentication method. With Basic Authentication, the username and password are included in the HTTP authorization header, and the credentials are verified through user registry.

User registries that are configured in Cloud Manager have the following characteristics:

- They are available to Cloud Manager, API Manager, the Developer Portal, and for Basic Authentication for APIs.
- They are available to provider organizations, as determined by the visibility setting.
- They can be edited and deleted only in Cloud Manager.

User registries that are configured in API Manager have the following characteristics:

- They are available to the Developer Portal (for Catalogs) and for Basic Authentication for APIs.
- They are available only to the Provider Organization that created them.

## User registries supported by API Connect

---

API Connect integrates with several types of user registries to accommodate all security solutions. You can use your corporate LDAP registry, an Authentication URL, or an LUR to provide secure access to Cloud Manager, API Manager, the Developer Portal, and APIs. API Connect includes two internal databases that serve as local user registries, or LURs. The Providers LUR supports credentials for Provider organizations and the Admin LUR supports credentials for the Administrator organization. Multiple registries may be configured.

The following user registry types can serve as a resource in API Connect:

- Local user registry (LUR) - An internal database of usernames and passwords stored on the local server. Contains the Admin user account which may not be modified or deleted.
- LDAP - A user registry definition that points to an existing corporate LDAP directory. May be set up as case-sensitive.
- Authentication URL - Accesses a URL that points to a service for validating user credentials.
- OpenID Connect (OIDC).
- [Configuring an Authentication URL user registry](#)  
An Authentication URL user registry provides a simple mechanism for authenticating users by referencing a custom identity provider.
- [Configuring a shared custom user registry](#)  
You can configure a custom user registry to provide user authentication for the Cloud Manager, the API Manager, and the Developer Portal.
- [Configuring an LDAP user registry in the Cloud Manager](#)  
You can use the Cloud Manager UI to configure an LDAP user registry as a shared resource to provide user authentication for the Cloud Manager, the API Manager, and the Developer Portal. APIs can also be secured with an LDAP user registry.
- [Configuring a Local User Registry](#)  
A Local User Registry (LUR) can be configured to provide user authentication for both Cloud Manager and API Manager.
- [Configuring an OIDC user registry](#)  
Configure a shared OIDC user registry for user onboarding and authentication when multi-factor authentication (MFA) is required.
- [Setting visibility for a user registry](#)  
The visibility setting determines which provider organizations can access a resource, for example, user registries.
- [Deleting a user registry](#)  
As the Cloud Manager administrator, you can delete user registries.
- [Removing a user from a user registry](#)  
As an API Connect user, you can remove yourself from an API Connect user registry. As an administrative user, you can remove other users from an API Connect user registry.

## Related tasks

---

- [Configuring an Authentication URL user registry](#)
- [Configuring a Local User Registry](#)
- [Configuring an LDAP user registry in the Cloud Manager](#)
- [Setting visibility for a user registry](#)

## Configuring an Authentication URL user registry

---

An Authentication URL user registry provides a simple mechanism for authenticating users by referencing a custom identity provider.

### About this task

---

This topic describes how to configure a new Authentication URL user registry as a Resource in your cloud. After the user registry is configured, you must select it for use in your cloud in Settings > User Registries. See [Selecting user registries for Cloud Manager and API Manager](#).

One of the following roles is required to configure user registries.:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

Note:

API Connect issues an HTTP `GET` call to the Authentication URL endpoint, sending the user's credential. The following example shows a call made to an Authentication URL identity provider with an endpoint defined as `https://myauthurl.example.com/user/authenticate`:

```
GET /user/authenticate HTTP/1.1
Host: myauthurl.example.com
Authorization: Basic c3Bvb246Zm9yaw=
```

If the Authentication URL endpoint returns an HTTP status code of `200`, the user authenticates successfully. An HTTP status code other than `200` indicates a failed login attempt. API Connect forwards any HTTP Header starting with `x-` (with the exception of `x-Client-Certificate`), and Cookie to the Authentication URL identity

provider, to aid the authentication decision; for example:

```
GET /user/authenticate HTTP/1.1
Host: myauthurl.example.com
Authorization: Basic c3Bvb246Zm9yaw=
X-Forwarded-For: 8.8.9.9
X-Custom-Header-From-Customer: special
Cookie: MyCookie=VGhpc0lzV21ja2VkQW1hemluZw==
```

When a user is presented with the form for completing their API Connect user registration, which fields are pre-populated depends on which fields are returned in the response from the Authentication URL identity provider. If any of the following fields are returned, they will be pre-populated in the registration form:

- `username`
- `email`
- `first_name`
- `last_name`

If the `username` field is not returned, the registration form displays the user name that was provided by the user. The pre-population capability requires that the response from the Authentication URL identity provider satisfies the following conditions:


- The `Content-Type` must be `application/json`.
- The response body format must be JSON.

A sample response is as follows:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
```

```
{
 "username": "myuser",
 "email": "myuser@example.com",
 "first_name": "My",
 "last_name": "User"
}
```

## Procedure

1. In the Cloud Manager, click .
2. Select User Registries to see the list of current user registries in your cloud.
3. Click Create in the User Registries section.

Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.

4. Select Authentication URL User Registry and enter the following parameters:

Field	Description
Title (required)	Enter a descriptive name to use on the screen.
Name (required)	The name that is used in CLI commands. The name is auto-generated. For details of the CLI commands for managing user registries, see <a href="#">apic user-registries</a> .
Summary (optional)	Enter a brief description.
Display Name (required)	The name that is displayed for selection by the user when logging in to a user interface, or activating their API Manager account. For details of user interface log in, and account activation, see <a href="#">Accessing the Cloud Manager user interface</a> , <a href="#">Accessing the API Manager user interface</a> , and <a href="#">Activating your API Manager user account</a> . Note: The Developer Portal uses the <b>Title</b> of the User Registries when rendering them at the login page, rather than the <b>Display Name</b> .
URL (required)	Enter the URL for the authentication service. When establishing authentication, API Connect makes a GET call to the URL. The call includes the user name and password it has collected from the user in its authorization header. Either 200 OK or 401 Unauthorized will be returned.
TLS Client Profile (optional)	Select a TLS Client Profile to allow secure authentication with a specific web server.
Case sensitive	To ensure proper handling of user name capitalization, you <b>must</b> ensure that your case-sensitivity setting here matches the setting on your backend Authentication URL server: <ul style="list-style-type: none"><li>• <b>Only</b> select Case sensitive if your backend server supports case-sensitivity.</li><li>• <b>Do not</b> select Case sensitive if your backend server does not support case-sensitivity.</li></ul> Note: The Developer Portal does not support case sensitive usernames. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
Email required	Select this checkbox if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this checkbox if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

5. Click Save.

## Results

The Authentication URL user registry is saved and is available to be selected as a user registry for Cloud Manager or API Manager by selecting it in Settings > User Registries. It can also be used for Basic Authentication in the Security Definition for an API.

## Related concepts

- [Managing authentication and security](#)
- [User registries overview](#)

## Related tasks

- [Configuring a Local User Registry](#)
- [Configuring an LDAP user registry in the Cloud Manager](#)
- [Setting visibility for a user registry](#)

## Configuring a shared custom user registry

You can configure a custom user registry to provide user authentication for the Cloud Manager, the API Manager, and the Developer Portal.

### Before you begin

To configure a custom user registry as a resource in the Cloud Manager, the external user directory must be created and available to use with your API Connect ecosystem.

Custom user registries can be used for authenticating users to the Cloud Manager, the API Manager, and the Developer Portal, but cannot be used to secure APIs.

One of the following roles is required to configure a custom user registry:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage permissions`

### About this task

You can create a custom user registry that is specific to a provider organization, or one that can be shared and available to all of the provider organizations in your API Connect environment. An organization-specific custom user registry can be used for authenticating Developer Portal users, while a shared custom user registry can be used for authenticating Cloud Manager, API Manager, and Developer Portal users.

This topic describes how to configure a shared custom user registry. If you want to create an organization-specific registry, see [Creating an organization-specific custom user registry](#) for more information. Note also that the visibility of a user registry is set to **shared** by default. However, you can change the visibility setting to make the registry private, or visible only to specific provider organizations. For more information, see [Setting visibility for a user registry](#).

Note:

- You can also create custom user registries by using the API Connect REST APIs (see the [API Connect REST API documentation](#)).

You can use the following instructions to create a writable or a read-only custom user registry.

API Connect provides two methods for creating a shared custom user registry in the Cloud Manager, as described in the following sections:


- [Creating a shared custom user registry by using the Cloud Manager UI](#)
- [Creating a shared custom user registry by using the developer toolkit CLI](#)

## Creating a shared custom user registry by using the Cloud Manager UI

You configure a custom user registry by first creating a custom user registry type that defines your external user registry, and then creating a new custom user registry resource that references the custom user registry type that you just created. To make the registry available for user authentication, you need to add the registry to the Cloud Manager, or to the API Manager, or both, depending on your requirements. When the custom user registry is used for authentication, API Connect makes a REST call to the endpoint of your external registry, as defined in the custom user registry type.

## Creating a custom user registry type

Use the following instructions to create a custom user registry type that defines your external user registry.

1. In the Cloud Manager, click  Resources.
2. Click Create in the User registry type section.
3. Enter the following information:

Property	Description
Title	A descriptive name of the custom user registry type to display in the UI. For example, <i>My custom user registry type</i> .
Name	The name of the user registry type. This name is auto-generated, and is used in CLI commands.
Version	The version number of the custom user registry type.
Summary	A brief description of the custom user registry type.
Properties	Enter the properties of your external user registry. Must include the property: <code>remote: true</code> This property must be set to <code>true</code> to indicate that this is an external user registry.

Property	Description
Secured Endpoint	Denotes the external registry secured endpoint section of the custom user registry type.
URL	The secure endpoint of your external user registry interface, where your platform REST API is located. It is made up of the URL and port, for example:  <code>https://custom.ms.com:8888</code>  API Connect will make REST calls to this endpoint for the user registry actions.
TLS Client Profile	Optionally set the TLS Client Profile URL that the custom server requires.
Headers	Optionally complete the user-defined header section to your external registry specification.
Configuration schema	Optionally complete the user-defined configuration schema section to your external registry specification.

4. Click Create. A new user registry with a Type of Custom is displayed in the User registry type list.

## Creating a custom user registry resource

Use the following instructions to create a new custom user registry resource that references the custom user registry type that you just created.

1. Click Create in the User Registries section of Resources.
2. Click the Custom user registry tile.
3. Enter the following information:

Property	Description
Custom type	The name of the custom user registry type that you just created. For example, <i>my-custom-user-registry-type</i> . Use the drop-down arrow to select different custom user registry types.
Title	A descriptive name to display in the UI.
Name	The name of the custom user registry. This name is auto-generated, and is used in CLI commands.
Summary	A brief description of the custom user registry.
Endpoint	Optional endpoint information.
Case sensitive	To ensure proper handling of user name capitalization, you <b>must</b> ensure that your case-sensitivity setting here matches the setting on your backend server: <ul style="list-style-type: none"> <li>• <b>Only</b> select Case sensitive if your backend server supports case-sensitivity.</li> <li>• <b>Do not</b> select Case sensitive if your backend server does not support case-sensitivity.</li> </ul> Note: <ul style="list-style-type: none"> <li>• The Developer Portal does not support case sensitive usernames.</li> <li>• After at least one user has been onboarded into the registry, you cannot change this setting.</li> </ul>
User registry managed	Determines whether API Connect manages your user registry. Valid values are: <ul style="list-style-type: none"> <li>• <b>true</b> - select the checkbox</li> <li>• <b>false</b> - clear the checkbox</li> </ul>
User managed	Determines whether your user registry is writable or not. Select the checkbox to set to <b>true</b> for writable. Clear the checkbox for the non-writable option.
Email required	Select this checkbox if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this checkbox if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

4. Click Create. The new user registry is displayed in the User registries list.

## Adding your custom user registry to the Cloud Manager or API Manager login

To make the custom user registry available for user authentication in the Cloud Manager or the API Manager, you must set it as active in the Settings > User Registries section of the Cloud Manager UI. See [Selecting user registries for Cloud Manager and API Manager](#) for more information.

If you want to make the custom user registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. In the API Manager UI, click Manage followed by the relevant Catalog, and then click Catalog settings > Onboarding. In the Catalog User Registries section, click Edit, select the user registry, and click Save. For more information, see [Creating and configuring Catalogs](#).

## Creating a shared custom user registry by using the developer toolkit CLI

You configure a custom user registry by first configuring an integration document that defines your external user registry, and then creating a new custom user registry resource that references the integration document that you just created. You use developer toolkit CLI commands to create the integration document, and then the custom user registry. Finally, to make the registry available for user authentication, you need to configure the registry in the management server, or in the appropriate Catalog, or both, depending on your requirements. When the custom user registry is used for authentication, API Connect makes a REST call to the endpoint of your external registry, as defined in the integration document.

## Logging in to the management server CLI

Before you can define the custom user registry, you must log in to your management server from the developer toolkit CLI as a member of the cloud administration organization. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm admin/identity_provider
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **default-idp-1**. For full details of the login command, see [Logging in to a management server](#).

For more information about how to use the CLI, see [Installing the toolkit](#), and [Overview of the command-line tool](#).

## Defining your integration document

You define the configuration of your external user registry in an **integration\_file.yaml** document, as shown in the following example.

```
integration: '1.0.0'
integration_type: 'user_registry'
info:
 name: 'custom_user_registry'
 version: '1.0.0'
 title: 'Custom External User Registry'
 summary: 'Configure External User Registry'
properties:
 remote: true
secured_endpoint:
 endpoint: 'https://custom.ms.com:8888'
 tls_client_profile_url: {$tlsClientProfileUrl}
headers:
 key1: value1
 key2: value2
configuration_schema:
 option_config1:
 type: string
 option_config2:
 type: boolean
```

The integration document properties are described in the following table:

Property	Description
<b>integration_type</b>	The type of integration, which in this case is <b>user_registry</b> .
<b>info</b>	Denotes the information section for the integration document.
<b>name</b>	The name of the user registry type, for example <b>custom_user_registry</b> . This name is used in the user registry file to denote the registry type.
<b>version</b>	The version number of the integration document.
<b>title</b>	A descriptive name to display in a graphical user interface.
<b>summary</b>	A brief description of the integration document.
<b>properties</b>	Denotes the property section of the integration document.
<b>remote</b>	Indicates that this is an external user registry. Must be set to <b>true</b> .
<b>secured_endpoint</b>	Denotes the external registry endpoint section of the integration document.
<b>endpoint</b>	The secure endpoint of your external user registry interface, where your platform REST API is located. It is made up of the url and port, for example:  <b>https://custom.ms.com:8888</b>  API Connect will make REST calls to this endpoint for the user registry actions.
<b>tls_client_profile_url</b>	Optionally set the TLS Client Profile URL that the custom server requires.
<b>headers</b>	Complete the user-defined header section to your external registry specification.
<b>configuration_schema</b>	Complete the user-defined configuration schema section to your external registry specification.

Save your **integration\_file.yaml** so it can be accessed by the **integrations:create** command in the following section.

## Creating your custom integration

To create your custom user registry integration, run the following CLI command:

```
apic integrations:create --server mgmt_endpoint_url --subcollection user-registry integration_file.yaml
```

where:

- **mgmt\_endpoint\_url** is the management platform API endpoint URL.
- **integration\_file** is the name of the YAML file that defines the configuration of your external user registry.

At this point API Connect makes a REST request call to the **endpoint** that is configured in the integration document, and the external platform REST API will need to return the requested information.

On completion of the integration creation, the command displays the following summary details:

```
integration_name custom_integration_url
```

The **integration\_name** is derived from the **name** property in the configuration YAML file. The **custom\_integration\_url** is the URL with which the integration can be accessed. Both properties are needed when defining your custom user registry in the next section.

After your custom integration has completed successfully, you can continue to create the custom user registry.

## Defining your custom user registry

You define the configuration of your custom user registry in a `custom_config_file.yaml` file, as shown in the following example.

```
name: 'custom_registry_name'
title: 'display_title'
registry_type: 'custom_user_registry'
integration_url: custom_integration_url
case_sensitive: true_or_false
user_managed: true_or_false
user_registry_managed: true_or_false
email_required: true_or_false
email_unique_if_exist: true_or_false
identity_providers:
- name: provider_name
 title: 'provider_title'
configuration:
 custom_config1: 'value1'
 customize: true
```

The registry properties are described in the following table:

Property	Description
<code>name</code>	The name of the custom user registry. This name is used in the CLI commands.
<code>title</code>	A descriptive name to display in a graphical user interface.
<code>registry_type</code>	The registry type that you configured in the <code>name</code> property in the integration document, for example <code>custom_user_registry</code> .
<code>integration_url</code>	The custom integration URL in your API Connect configuration. You can determine the custom integration URL by using the following CLI command:  <code>apic integrations:list --server mgmt_endpoint_url --subcollection user-registry</code>  Or you can copy and paste from the summary of the integration creation.
<code>case_sensitive</code>	Determines whether your user registry is case-sensitive. Valid values are: <ul style="list-style-type: none"> <li><code>true</code></li> <li><code>false</code></li> </ul> <p>To ensure proper handling of user name capitalization, you <b>must</b> ensure that your case-sensitivity setting here matches the setting on your backend LDAP server:</p> <ul style="list-style-type: none"> <li><b>Only</b> set <code>case_sensitive</code> to <code>true</code> if your backend LDAP server supports case-sensitivity.</li> <li>Set <code>case_sensitive</code> to <code>false</code> if your backend LDAP server does not support case-sensitivity.</li> </ul> <p>Note: After at least one user has been onboarded into the registry, you cannot change this setting.</p>
<code>user_managed</code>	Determines whether your user registry is writable or not. Must be set to <code>true</code> for writable. Set to <code>false</code> if you don't want the registry to be writable.
<code>user_registry_managed</code>	Determines whether API Connect manages your user registry. Valid values are: <ul style="list-style-type: none"> <li><code>true</code></li> <li><code>false</code></li> </ul>
<code>email_required</code>	Determines whether an email address is required as part of the user onboarding process. Valid values are: <ul style="list-style-type: none"> <li><code>true</code></li> <li><code>false</code></li> </ul> <p>If set to <code>true</code>, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.</p>
<code>email_unique_if_exist</code>	Determines whether email addresses must be unique within the user registry. Valid values are: <ul style="list-style-type: none"> <li><code>true</code></li> <li><code>false</code></li> </ul> <p>Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.</p>
<code>identity_providers</code>	An array containing the details of your custom server, where: <ul style="list-style-type: none"> <li><code>name</code> - is the name of the custom server and is the name that is used in CLI commands</li> <li><code>title</code> - is the display name of the custom server</li> </ul>
<code>configuration</code>	The user-defined configuration based on the <code>configuration_schema</code> that is defined in the integration document.

Save your `custom_config_file.yaml` so it can be accessed by the `user-registries:create` command in the following section.

## Creating your custom user registry

To create your shared custom user registry, run the following CLI command:

```
apic user-registries:create --server mgmt_endpoint_url --org admin custom_config_file.yaml
```

where:

- `mgmt_endpoint_url` is the platform API endpoint URL.



- `--org admin` means that the registry will be created in the admin organization, which is required for a user registry to be shared.
- `custom_config_file` is the name of the YAML file that defines the configuration of your custom user registry.

On completion of the registry creation, the command displays the following summary details:

```
registry_name registry_url
```

The `registry_name` is derived from the `name` property in the custom user registry YAML file. The `registry_url` is the URL with which the custom registry resource can be accessed.

Your shared custom user registry is now created; see the following sections for instructions on how to make the registry available to users.

## Configuring your custom registry for Cloud Manager or API Manager login

If you want to make your custom registry available for authenticating Cloud Manager and API Manager users, you must configure it on the management server.

1. Determine the URL of your custom user registry by using the following command (or you can copy and paste from the summary of the registry creation):

```
apic user-registries:list --server mgmt_endpoint_url --org admin
```

2. Determine what the current user registries are, because you will need to confirm these as well as add your new custom registry:

```
apic user-registry-settings:get --server mgmt_endpoint_url --output -
```

This command outputs a list of all the current user registries in your environment, similar to the following example:

```
type: user_registry_setting
api_version: 2.0.0
name: user-registry-setting
admin_user_registry_default_url: >-
 https://server.com/api/user-registries/xxxxx/xxxxx-1234
admin_user_registry_urls:
 - >-
 https://server.com/api/user-registries/xxxxx/xxxxx-1234
provider_user_registry_default_url: >-
 https://https://server.com/api/user-registries/xxxxx/xxxxx-5678
provider_user_registry_urls:
 - >-
 https://https://server.com/api/user-registries/xxxxx/xxxxx-5678
created_at: '2019-09-30T12:22:19.467Z'
updated_at: '2019-10-17T10:05:37.867Z'
url: 'https://server.com/api/cloud/settings/user-registries'
```

3. Enter the following command to update your user registries (the terminating hyphen character means that the command takes input from the command line):

```
apic user-registry-settings:update --server mgmt_endpoint_url -
```

The following message is output:

```
Reading USER_REGISTRY_SETTING_FILE arg from stdin
```

4. If you want to make your custom registry available for authenticating Cloud Manager users, enter the following data, followed by a new line:

```
admin_user_registry_urls:
 - >-
 current_admin_user_registry_urls
 - new_custom_user_registry_url
```

where:

- `current_admin_user_registry_urls` are the current admin user registry URLs, as listed in Step 2 in the `admin_user_registry_urls` section. Note that you must include all of the user registry URLs that you want to remain, listing each URL on a new line.
- `new_custom_user_registry_url` is the URL of your new custom user registry, as determined in Step 1.

5. If you want to make your custom registry available for authenticating API Manager users, enter the following data, followed by a new line:

```
provider_user_registry_urls:
 - >-
 current_provider_user_registry_urls
 - new_custom_user_registry_url
```

where:

- `current_provider_user_registry_urls` are the current provider organization user registry URLs, as listed in Step 2 in the `provider_user_registry_urls` section. Note that you must include all of the user registry URLs that you want to remain, listing each URL on a new line.
- `new_custom_user_registry_url` is the URL of your new custom user registry, as determined in Step 1.

6. Press `CTRL D` to terminate the input. A confirmation message is output, for example:

```
user-registry-setting https://server.com/api/cloud/settings/user-registries
```

where `server.com` is your management server endpoint.

## Configuring your custom registry in a Catalog

If you want to make your custom registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. Complete the following steps:

1. Determine the URL of your custom user registry by using the following command (or you can copy and paste from the summary of the registry creation):

```
apic user-registries:list --server mgmt_endpoint_url --org admin
```

2. Log in to the management server as a member of a provider organization by entering the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

For full details of the `apic login` command, see [Logging in to a management server](#).

3. Enter the following command (the terminating hyphen character means that the command takes input from the command line):

```
apic configured-catalog-user-registries:create --server mgmt_endpoint_url --org organization_name --catalog catalog_name -
```

where `catalog_name` is the value of the `name` property of the required Catalog. The command returns:

```
Reading CONFIGURED_CATALOG_USER_REGISTRY_FILE arg from stdin
```

4. Enter the following data, followed by a new line:

```
user_registry_url: custom_registry_url
```

where `custom_registry_url` is the URL of your custom registry, obtained in step 1.

5. Press **CTRL D** to terminate the input.

## Related information

- [Command-line tool reference for the developer toolkit](#)

## Configuring an LDAP user registry in the Cloud Manager

You can use the Cloud Manager UI to configure an LDAP user registry as a shared resource to provide user authentication for the Cloud Manager, the API Manager, and the Developer Portal. APIs can also be secured with an LDAP user registry.

### Before you begin

To configure an LDAP user registry as a shared resource in the Cloud Manager, the LDAP directory must be created for use with your API Connect ecosystem.

LDAP registries can be used to secure APIs, to authenticate users to the Cloud Manager and the API Manager, or for securing a Catalog to authenticate Developer Portal users.

Important: If you are using an LDAP registry to secure APIs, the STARTTLS protocol, which upgrades an insecure protocol to a secure one by applying TLS security, is not supported.

One of the following roles is required to configure an LDAP user registry:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage permissions`

### About this task

You can create an LDAP user registry that is specific to a provider organization, or one that can be shared and available to all of the provider organizations in your API Connect environment. An organization-specific LDAP user registry can be used for authenticating Developer Portal users in a specific provider organization. While a shared LDAP user registry can be used across the Cloud Manager, the API Manager, and the Developer Portal components in your environment.

This topic describes how to configure a shared LDAP user registry that is available to all of the provider organizations in your API Connect environment. If you want to create an organization-specific registry, see [Creating an LDAP user registry in API Manager](#) for more information. Note also that the visibility of a user registry is set to **shared** by default. However, you can change the visibility setting to make the registry private, or visible only to specific provider organizations. For more information, see [Setting visibility for a user registry](#).

Note:


- If you configure your LDAP user registry to be writable (by selecting the User Managed check box on the registry), you can use the Developer Portal UI for onboarding and authenticating new Developer Portal users, as well as those users that already exist in the LDAP database. A writable LDAP user registry cannot be used to authenticate Cloud Manager and API Manager users.
- You can also create and manage LDAP user registries by using the developer toolkit CLI (see [Using the CLI to configure a shared LDAP user registry](#)), and by using the API Connect REST APIs (see the [API Connect REST API documentation](#)).
- If you are using the DataPower® API Gateway, LDAP group authentication is not supported.
- You can map external LDAP groups to API Connect user roles to enable greater control of user access, but this configuration can be done only by using the developer toolkit CLI; see [Using the CLI to configure a shared LDAP user registry](#) for details.

You create an LDAP user registry by configuring a set of properties in the Cloud Manager UI. If you want to enable writable LDAP, you must complete the Attribute Mapping section by selecting the User Managed checkbox, and providing the mapping of your source LDAP attribute names to the target API Connect values. You can also change a registry to be read-only again by clearing the User Managed checkbox. After configuring the user registry, you must set it as active in Settings > User Registries. To make the registry available to the Developer Portal, you must define the registry for consumer onboarding in the associated Catalog. To secure APIs with an LDAP registry, you must configure security definitions.

For general information about authenticating with LDAP, see [LDAP authentication](#).

## Procedure

Follow these steps to configure a new LDAP user registry as a shared resource in the Cloud Manager UI.

1. In the Cloud Manager, click  Resources.
2. Click Create in the User Registries section.

Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the

Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.

3. Select LDAP User Registry for the user registry type, and enter the following information:

Field	Description
Title	Enter a descriptive name to display on the screen.
Name	The name that is used in CLI commands. The name is auto-generated. For details of the CLI commands for managing user registries, see <a href="#">apic user-registries</a> .
Display Name (required)	The name that is displayed for selection by the user when logging in to a user interface, or activating their API Manager account. For details of user interface log in, and account activation, see <a href="#">Accessing the Cloud Manager user interface</a> , <a href="#">Accessing the API Manager user interface</a> , and <a href="#">Activating your API Manager user account</a> .  Note: The Developer Portal uses the <b>Title</b> of the User Registries when rendering them at the login page, rather than the <b>Display Name</b> .
Summary (optional)	Enter a brief description.
Address	Enter the IP address or host name of the LDAP server.
Port	Enter the Port number that API Connect can use to communicate with the LDAP registry. For example, 389.
Select a TLS Client Profile (optional)	Select the TLS Client Profile that the LDAP server requires.
Select an LDAP protocol version	Select the version number for the LDAP protocol that you are using.
Case sensitive	To ensure proper handling of user name capitalization, you <b>must</b> ensure that your case-sensitivity setting here matches the setting on your backend LDAP server: <ul style="list-style-type: none"> <li>• <b>Only</b> select Case sensitive if your backend LDAP server supports case-sensitivity.</li> <li>• Do <b>not</b> select Case sensitive if your backend LDAP server does not support case-sensitivity.</li> </ul> Note: The Developer Portal does not support case sensitive usernames. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
Enable External Group Mapping	Enable this property if you want user to use this user registry to map external LDAP groups to API Connect user roles to enable more control of user authorization.
Email required	Select this check box if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this check box if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

4. Click Next and enter the authentication information, which will vary depending on the selected Authentication Method. The choices are:

- Compose DN - Select this format if you can compose the user LDAP Distinguished Name (DN) from the user name. For example, `uid=<username>,ou=People,dc=company,dc=com` is a DN format that can be composed from the user name. If you are unsure whether Compose (DN) is the correct option, contact your LDAP administrator. If you are using an LDAP registry to secure APIs, Compose DN is not supported with the DataPower API Gateway.
- Compose UPN - Select this format if your LDAP directory supports binding with User Principal Names such as `john@acme.com`. The Microsoft Active Directory is an example of an LDAP directory that supports Compose UPN authentication. If you are unsure whether your LDAP directory supports binding with UPNs, contact your LDAP administrator.  
Note: The Admin Bind DN and Admin Bind Password are not used with this authentication method.
- Search DN - Select this format if you cannot compose the user LDAP Distinguished Name from the user name; for example, if the base DNs of the users are different. This format might require an administrator DN and password to search for users in the LDAP directory. If your LDAP directory permits anonymous binds, you can omit the admin DN and password. If you are unsure if your LDAP directory permits anonymous binds, contact your LDAP administrator.

For all of the authentication methods:

If you are creating an LDAP registry to authenticate users of an API, you can specify an LDAP authorization group to restrict API access. To be able to call an API that is secured by the LDAP registry, a user must successfully authenticate with their LDAP user ID and password **and** they must be a member of the specified authorization group. The authorization group can be a Static Group or Dynamic Group. A static group is one in which the individual members of the group are explicitly listed. A dynamic group is one which is defined according to the set of attributes that the group members share in common.

5. For authentication method Compose DN, enter the following:

Field	Description
Bind Method	Anonymous or Authenticated. If specific permissions are not needed to search the registry, select Anonymous Bind. Or, if specific permissions are necessary, select Authenticated Bind.
Admin DN	For Authenticated Bind, enter the Distinguished Name of a user authorized to perform searches in the LDAP directory. For example <code>cn=admin,dc=company,dc=com</code> .
Admin Password	For Authenticated Bind, enter the user password for the Admin DN.
Prefix	Specify the prefix to the DN. For example (uid=.
Suffix	Specify the suffix to the DN. For example ).
Base DN (optional)	Enter a base DN in the Base DN field, or click Get Base DN to populate the field with a retrieved base DN.
Use group authentication (optional)	Static or Dynamic. For Static Group, enter the Group Based DN, Prefix, and Suffix. For Dynamic Group, enter the Filter condition for the group.

6. For authentication method Compose UPN, enter the following:

Field	Description
Bind Method	Anonymous or Authenticated. If specific permissions are not needed to search the registry, select Anonymous Bind. Or, if specific permissions are necessary, select Authenticated Bind.
Admin DN	For Authenticated Bind, enter the Distinguished Name of a user authorized to perform searches in the LDAP directory. For example <code>cn=admin,dc=company,dc=com</code> .
Admin Password	For Authenticated Bind, enter the user password for the Admin DN.
Suffix	Enter the domain part of the user principal name. For example, <code>@acme.com</code> .
Use group authentication (optional)	Enter the Filter condition for the group.

7. For authentication method Search DN, enter the following:

Field	Description
Bind Method	Anonymous or Authenticated. If specific permissions are not needed to search the registry, select Anonymous Bind. Or, if specific permissions are necessary, select Authenticated Bind.
Admin DN	For Authenticated Bind, enter the Distinguished Name of a user authorized to perform searches in the LDAP directory. For example <code>cn=admin,dc=company,dc=com</code> .
Admin Password	For Authenticated Bind, enter the user password for the Admin DN.
Prefix	Specify the prefix to the DN. For example <code>(uid=</code> .
Suffix	Specify the suffix to the DN. For example <code>)</code> .
Base DN (optional)	Enter a base DN in the Base DN field, or click Get Base DN to populate the field with a retrieved base DN.
Use group authentication (optional)	Static or Dynamic. For Static Group, enter the Group Based DN, Prefix, and Suffix. For Dynamic Group, enter the Filter condition for the group.

8. Optional: Click Test configuration to test the settings for your LDAP user registry. Enter valid credentials to ensure that you can access the LDAP database.
9. Optional: If you want to make your LDAP user registry writable, select the User Managed checkbox in the Attribute Mapping section, and provide the mapping of your source LDAP attribute names to the target API Connect values. Click Add to add each name/value pair, specified as follows:
- LDAP ATTRIBUTE NAME - is the name of the source LDAP attribute.
  - API CONNECT VALUE - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in [ ] with the value that the user supplies when signing up.

The default user profile properties that API Connect requires during user registration are `username`, `first_name`, `last_name`, `email`, and `password`, as shown in the following example:

LDAP ATTRIBUTE NAME	API CONNECT VALUE
<code>dn</code>	<code>uid={username},ou=users,dc=company,dc=com</code>
<code>cn</code>	<code>[first_name] [last_name]</code>
<code>sn</code>	<code>[last_name]</code>
<code>mail</code>	<code>[email]</code>
<code>userPassword</code>	<code>[password]</code>

You must ensure that you enter the correct attribute mapping values for your LDAP configuration, to enable API Connect to access the LDAP database. Note that a writable LDAP user registry cannot be used to authenticate Cloud Manager and API Manager users.

10. Click Create.
- Your new LDAP registry is shown in the list of User Registries on the Resources page.

## What to do next

To make the LDAP registry available for user authentication in the Cloud Manager and the API Manager, you must set it as active in the Settings > User Registries section. See [Selecting user registries for Cloud Manager and API Manager](#) for more information.

If you want to make the LDAP registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. In the API Manager UI, click Manage followed by the relevant Catalog, and then click Settings > Onboarding. In the Catalog User Registries section, click Edit, select the user registry, and click Save. For more information, see [Creating and configuring Catalogs](#).

If you want to use the LDAP user registry to secure APIs, see the following information:

- To use for basic authentication in the security definition for an API, see [Creating a basic authentication security definition](#).
- To use for authentication in the User Security configuration for a native OAuth provider, see [Configuring user security for a native OAuth provider](#).
- Using the CLI to configure a shared LDAP user registry**  
You can use the developer toolkit CLI to configure an LDAP user registry to provide user authentication for the Cloud Manager, the API Manager, and the Developer Portal. APIs can also be secured with an LDAP user registry.

## Related concepts

- [Managing authentication and security](#)
- [User registries overview](#)

## Related tasks

- [Configuring a Local User Registry](#)
- [Configuring an Authentication URL user registry](#)
- [Setting visibility for a user registry](#)

## Related information

- [LDAP authentication](#)
- [Securing your API Connect Cloud with LDAP \(series of developer articles\)](#)

## Using the CLI to configure a shared LDAP user registry

You can use the developer toolkit CLI to configure an LDAP user registry to provide user authentication for the Cloud Manager, the API Manager, and the Developer Portal. APIs can also be secured with an LDAP user registry.

## Before you begin

To configure an LDAP user registry as a resource in the Cloud Manager, the LDAP directory must be created and available to use with your API Connect ecosystem.

LDAP registries can be used to secure APIs, or for authenticating users to the Cloud Manager, the API Manager, and the Developer Portal.

Important: If you are using an LDAP registry to secure APIs, the STARTTLS protocol, which upgrades an insecure protocol to a secure one by applying TLS security, is not supported.

One of the following roles is required to configure an LDAP user registry:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage permissions`

## About this task

You can create an LDAP user registry that is specific to a provider organization, or one that can be shared and available to all of the provider organizations in your API Connect environment. An organization-specific LDAP user registry can be used for onboarding and authenticating Developer Portal users, while a shared LDAP user registry can be used for authenticating Cloud Manager, API Manager, and Developer Portal users.

This topic describes how to configure a shared LDAP user registry. If you want to create an organization-specific registry, see [Using the CLI to create an organization-specific LDAP user registry](#), for more information. Note also that the visibility of a user registry is set to `shared` by default. However, you can change the visibility setting to make the registry private, or visible only to specific provider organizations. For more information, see [Setting visibility for a user registry](#).

Note:

- If the LDAP user registry is configured as writable (by enabling the `user-managed` property on the registry), you can use the Developer Portal UI for onboarding and authenticating new Developer Portal users, as well as those users that already exist in the LDAP database. A writable LDAP user registry cannot be used to authenticate Cloud Manager and API Manager users.
- You can also create LDAP user registries by using the Cloud Manager UI (see [Configuring an LDAP user registry in the Cloud Manager](#)), and by using the API Connect REST APIs (see the [API Connect REST API documentation](#)).
- If you are using the DataPower® API Gateway, LDAP group authentication is not supported.
- You can map external LDAP groups to API Connect user roles to enable more control of user authorization by using the developer toolkit CLI. The following instructions explain how to set the `external_group_mapping_enabled` configuration on your LDAP user registry resource. For information about how to set external role mapping on your API Connect user roles, see [Configuring LDAP group mappings on Cloud Manager user roles](#).

You configure an LDAP user registry by first defining the registry details in a configuration file. You then use a developer toolkit CLI command to create the registry, passing the configuration file as a parameter. Finally, to make the registry available for user authentication, you need to configure the registry in the management server, or in the appropriate Catalog, or both, depending on your requirements. To secure APIs with an LDAP registry, you must configure security definitions. You can use the following instructions to create a writable or a read-only LDAP user registry.

For general information about authenticating with LDAP, see [LDAP authentication](#).

## Logging in to the management server CLI

Before you can define the LDAP user registry configuration, you must log in to your management server from the developer toolkit CLI as a member of the cloud administration organization. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm admin/identity_provider
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`.

For full details of the login command, see [Logging in to a management server](#).

For more information about how to use the CLI, see [Installing the toolkit](#), and [Overview of the command-line tool](#).

## Defining your LDAP configuration

You define the configuration of your LDAP user registry in an `ldap_config_file.yaml` file, as shown in the following example. Note that the actual contents of your YAML file will vary depending on the authentication method of your LDAP server, and this is explained in the following tables.

```
name: registry_name
title: "display_title"
integration_url: LDAP_integration_url
user_managed: true_or_false
user_registry_managed: false

external_group_mapping_enabled: true_or_false
case_sensitive: true_or_false
email_required: true_or_false
email_unique_if_exist: true_or_false
identity_providers:
- name: provider_name
 title: provider_title
endpoint:
```

```

endpoint: "ldap_server_url_and_port"
configuration:
 authentication_method: authentication_method
 authenticated_bind: "true_or_false"
 admin_dn: "admin_dn"
 admin_password: admin_password
 search_dn_base: "search_dn_base"
 search_dn_scope: search_dn_scope
 search_dn_filter_prefix: prefix
 search_dn_filter_suffix: suffix
 attribute_mapping:
 dn: "distinguished_name"
 cn: "common_name"
 sn: "last_name"
 mail: "email_address"
 userPassword: "password"

```

The registry properties that are common to each authentication method are described in the following table:

Property	Description
<b>name</b>	The name of the registry. This name is used in CLI commands.
<b>title</b>	A descriptive name to display in a graphical user interface.
<b>integration_url</b>	The LDAP integration URL in your API Connect configuration. You can determine the LDAP integration URL by using the following CLI command: <code>apic integrations:list --server mgmt_endpoint_url --subcollection user-registry</code>
<b>user_managed</b>	Determines whether your user registry is writable or not. Must be set to <b>true</b> for writable LDAP. You can change this setting to <b>false</b> if you don't want the registry to be writable; see the Switching your LDAP registry between writable and read-only section at the end of this topic for details. Note that a writable LDAP user registry cannot be used to authenticate Cloud Manager and API Manager users.
<b>user_registry_managed</b>	Must be set to <b>false</b> for LDAP. Determines whether API Connect manages your user registry. Only LUR registries are managed by API Connect.
<b>external_group_mapping_enabled</b>	Determines whether your user registry supports LDAP group mapping. Valid values are: <ul style="list-style-type: none"> <li><b>true</b></li> <li><b>false</b></li> </ul> <p>The default value is <b>false</b>.</p>
<b>case_sensitive</b>	Determines whether your user registry is case-sensitive. Valid values are: <ul style="list-style-type: none"> <li><b>true</b></li> <li><b>false</b></li> </ul> <p>To ensure proper handling of user name capitalization, you <b>must</b> ensure that your case-sensitivity setting here matches the setting on your backend LDAP server:</p> <ul style="list-style-type: none"> <li><b>Only</b> set <b>case_sensitive</b> to <b>true</b> if your backend LDAP server supports case-sensitivity.</li> <li>Set <b>case_sensitive</b> to <b>false</b> if your backend LDAP server does not support case-sensitivity.</li> </ul> <p>Note: After at least one user has been onboarded into the registry, you cannot change this setting.</p>
<b>email_required</b>	Determines whether an email address is required as part of the user onboarding process. Valid values are: <ul style="list-style-type: none"> <li><b>true</b></li> <li><b>false</b></li> </ul> <p>If set to <b>true</b>, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.</p>
<b>email_unique_if_exist</b>	Determines whether email addresses must be unique within the user registry. Valid values are: <ul style="list-style-type: none"> <li><b>true</b></li> <li><b>false</b></li> </ul> <p>Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.</p>
<b>identity_providers</b>	An array containing the details of your LDAP server, where: <ul style="list-style-type: none"> <li><b>name</b> - is the name of the LDAP server and is the name that is used in CLI commands</li> <li><b>title</b> - is the display name of the LDAP server</li> </ul>
<b>endpoint</b>	The endpoint of your LDAP server, made up of the url and port, for example: <code>"ldap://server.com:389"</code>
<b>tls_profile</b>	Optionally set the TLS Client Profile that the LDAP server requires.
<b>protocol_version</b>	Optionally set the version number for the LDAP protocol that you are using. Valid values are: <ul style="list-style-type: none"> <li>2</li> <li>3</li> </ul> <p>Defaults to 3 if not explicitly set.</p>

The properties in the configuration section will vary depending on the selected authentication method. The three authentication methods are:

- compose\_dn** - Set this format if you can compose the user LDAP Distinguished Name (DN) from the user name. For example, `uid=<username>,ou=People,dc=company,dc=com` is a DN format that can be composed from the user name. If you are unsure whether Compose (DN) is the correct option, contact your LDAP administrator. If you are using an LDAP registry to secure APIs, **compose\_dn** is not supported with the DataPower API Gateway.
- compose\_upn** - Set this format if your LDAP directory supports binding with User Principal Names such as `john@acme.com`. The Microsoft Active Directory is an example of an LDAP directory that supports Compose UPN authentication. If you are unsure whether your LDAP directory supports binding with UPNs, contact your LDAP administrator.

Note: The Admin Bind DN and Admin Bind Password are not used with this authentication method.

- **search\_dn** - Select this format if you cannot compose the user LDAP Distinguished Name from the user name; for example, if the base DN's of the users are different. This format might require an administrator DN and password to search for users in the LDAP directory. If your LDAP directory permits anonymous binds, you can omit the admin DN and password. If you are unsure if your LDAP directory permits anonymous binds, contact your LDAP administrator.

For authentication method **compose\_dn**, set the following configuration properties:

Properties	Description
<b>authentication_method</b>	<b>compose_dn</b>
<b>authenticated_bind</b>	<p>The bind method. Valid values are:</p> <ul style="list-style-type: none"> <li>• "true" - authenticated bind</li> <li>• "false" - anonymous bind</li> </ul> <p>If specific permissions are not needed to search the registry, select "false". If specific permissions are necessary, select "true".</p>
<b>admin_dn</b>	<p>If <b>authenticated_bind</b> is set to "true", enter the Distinguished Name (DN) of a user authorized to perform searches in the LDAP directory. For example:</p> <p>"cn=admin,dc=company,dc=com"</p>
<b>admin_password</b>	<p>If <b>authenticated_bind</b> is set to "true", enter the user password for the <b>admin_dn</b>.</p>
<b>search_dn_base</b>	<p>Optionally set a base DN, for example:</p> <p>"dc=company,dc=com"</p>
<b>bind_prefix</b>	<p>Set the prefix to the DN, for example:</p> <p>(uid=</p>
<b>bind_suffix</b>	<p>Set the suffix to the DN, for example:</p> <p>)</p>
<b>attribute_mapping</b>	<p>If <b>user_managed</b> is set to <b>true</b>, provide the mapping of your source LDAP attribute names to the target API Connect values. This mapping is configured as a name/value pair, specified as follows:</p> <p><b>ldap_registry_attribute_name: "apic_ldap_attribute_value"</b></p> <p>Where:</p> <ul style="list-style-type: none"> <li>• <b>ldap_registry_attribute_name</b> - is the name of the source LDAP attribute</li> <li>• <b>apic_ldap_attribute_value</b> - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in [ ] with the value that the user supplies when signing up.</li> </ul> <p>The user profile properties that API Connect requires during user registration are <b>username</b>, <b>first_name</b>, <b>last_name</b>, <b>email</b>, and <b>password</b>. The following extract shows an example of an attribute mapping:</p> <p><b>attribute_mapping:</b>  <b>dn: "uid=[username],ou=users,dc=company,dc=com"</b>  <b>cn: "[first_name] [last_name]"</b>  <b>sn: "[last_name]"</b>  <b>mail: "[email]"</b>  <b>userPassword: "[password]"</b></p>

For authentication method **compose\_upn**, set the following configuration properties:

Properties	Description
<b>authentication_method</b>	<b>compose_upn</b>
<b>authenticated_bind</b>	<p>The bind method. Valid values are:</p> <ul style="list-style-type: none"> <li>• "true" - authenticated bind</li> <li>• "false" - anonymous bind</li> </ul> <p>If specific permissions are not needed to search the registry, select "false". If specific permissions are necessary, select "true".</p>
<b>admin_dn</b>	<p>If <b>authenticated_bind</b> is set to "true", enter the Distinguished Name (DN) of a user authorized to perform searches in the LDAP directory. For example:</p> <p>"cn=admin,dc=company,dc=com"</p>
<b>admin_password</b>	<p>If <b>authenticated_bind</b> is set to "true", enter the user password for the <b>admin_dn</b>.</p>
<b>bind_suffix</b>	<p>Enter the domain part of the user principal name. For example:</p> <p>@acme.com</p>

Properties	Description
<b>attribute_mapping</b>	<p>If <code>user_managed</code> is set to <code>true</code>, provide the mapping of your source LDAP attribute names to the target API Connect values. This mapping is configured as a name/value pair, specified as follows:</p> <pre>ldap_registry_attribute_name: "apic_ldap_attribute_value"</pre> <p>Where:</p> <ul style="list-style-type: none"> <li><code>ldap_registry_attribute_name</code> - is the name of the source LDAP attribute</li> <li><code>apic_ldap_attribute_value</code> - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in [ ] with the value that the user supplies when signing up.</li> </ul> <p>The user profile properties that API Connect requires during user registration are <code>username</code>, <code>first_name</code>, <code>last_name</code>, <code>email</code>, and <code>password</code>. The following extract shows an example of an attribute mapping:</p> <pre>attribute_mapping: dn: "uid=[username],ou=users,dc=company,dc=com" cn: "[first_name] [last_name]" sn: "[last_name]" mail: "[email]" userPassword: "[password]"</pre>

For authentication method `search_dn`, set the following configuration properties:

Property	Description
<b>authentication_method</b>	<code>search_dn</code>
<b>authenticated_bind</b>	<p>The bind method. Valid values are:</p> <ul style="list-style-type: none"> <li><code>"true"</code> - authenticated bind</li> <li><code>"false"</code> - anonymous bind</li> </ul> <p>If specific permissions are not needed to search the registry, select <code>"false"</code>. If specific permissions are necessary, select <code>"true"</code>.</p>
<b>admin_dn</b>	<p>If <code>authenticated_bind</code> is set to <code>"true"</code>, enter the Distinguished Name (DN) of a user authorized to perform searches in the LDAP directory. For example:</p> <pre>"cn=admin,dc=company,dc=com"</pre>
<b>admin_password</b>	If <code>authenticated_bind</code> is set to <code>"true"</code> , enter the user password for the <code>admin_dn</code> .
<b>search_dn_base</b>	<p>Optionally set a base DN, for example:</p> <pre>"dc=company,dc=com"</pre>
<b>search_dn_scope</b>	<p>Optionally set the search DN scope. The scope determines which part of the directory information tree is examined. Possible values are:</p> <ul style="list-style-type: none"> <li><code>base</code></li> <li><code>one</code></li> <li><code>sub</code></li> </ul>
<b>search_dn_filter_prefix</b>	<p>Set the prefix to the DN, for example:</p> <pre>(uid=</pre>
<b>search_dn_filter_suffix</b>	<p>Set the suffix to the DN, for example:</p> <pre>)</pre>
<b>attribute_mapping</b>	<p>If <code>user_managed</code> is set to <code>true</code>, provide the mapping of your source LDAP attribute names to the target API Connect values. This mapping is configured as a name/value pair, specified as follows:</p> <pre>ldap_registry_attribute_name: "apic_ldap_attribute_value"</pre> <p>Where:</p> <ul style="list-style-type: none"> <li><code>ldap_registry_attribute_name</code> - is the name of the source LDAP attribute</li> <li><code>apic_ldap_attribute_value</code> - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in [ ] with the value that the user supplies when signing up.</li> </ul> <p>The user profile properties that API Connect requires during user registration are <code>username</code>, <code>first_name</code>, <code>last_name</code>, <code>email</code>, and <code>password</code>. The following extract shows an example of an attribute mapping:</p> <pre>attribute_mapping: dn: "uid=[username],ou=users,dc=company,dc=com" cn: "[first_name] [last_name]" sn: "[last_name]" mail: "[email]" userPassword: "[password]"</pre>

Save your `ldap_config_file.yaml` so it can be accessed by the `user-registries:create` command in the following section. See the [Example](#) section for an example configuration file.

## Creating your LDAP user registry

To create your shared LDAP user registry, run the following CLI command:

```
apic user-registries:create --server mgmt_endpoint_url --org admin ldap_config_file.yaml
```

where:

- `mgmt_endpoint_url` is the platform API endpoint URL.
- `--org admin` means that the registry will be created in the admin organization, which is required for a user registry to be shared.
- `ldap_config_file` is the name of the YAML file that defines the configuration of your LDAP user registry.



On completion of the registry creation, the command displays the following summary details:

```
registry_name registry_url
```

The `registry_name` is derived from the `name` property in the configuration YAML file. The `registry_url` is the URL with which the registry resource can be accessed. Your shared LDAP user registry is now created; see the following sections for instructions on how to make the registry available to users.

## Configuring your LDAP registry for Cloud Manager or API Manager login

If you want to make your LDAP registry available for authenticating Cloud Manager and API Manager users, you must configure it on the management server.

1. Determine the URL of your LDAP user registry by using the following command (or you can copy and paste from the summary of the registry creation):

```
apic user-registries:list --server mgmt_endpoint_url --org admin
```

2. Determine what the current user registries are, because you will need to confirm these as well as add your new LDAP registry:

```
apic user-registry-settings:get --server mgmt_endpoint_url --output -
```

This command outputs a list of all the current user registries in your environment, similar to the following example:

```
type: user_registry_setting
api_version: 2.0.0
name: user-registry-setting
admin_user_registry_default_url: >-
 https://server.com/api/user-registries/xxxxx/xxxxx-1234
admin_user_registry_urls:
- >-
 https://server.com/api/user-registries/xxxxx/xxxxx-1234
provider_user_registry_default_url: >-
 https://https://server.com/api/user-registries/xxxxx/xxxxx-5678
provider_user_registry_urls:
- >-
 https://https://server.com/api/user-registries/xxxxx/xxxxx-5678
created_at: '2019-09-30T12:22:19.467Z'
updated_at: '2019-10-17T10:05:37.867Z'
url: 'https://server.com/api/cloud/settings/user-registries'
```

3. Enter the following command to update your user registries (the terminating hyphen character means that the command takes input from the command line):

```
apic user-registry-settings:update --server mgmt_endpoint_url -
```

The following message is output:

```
Reading USER_REGISTRY_SETTING_FILE arg from stdin
```

4. If you want to make your LDAP registry available for authenticating Cloud Manager users, enter the following data, followed by a new line:

```
admin_user_registry_urls:
- >-
 current_admin_user_registry_urls
- new_ldap_user_registry_url
```

where:

- `current_admin_user_registry_urls` are the current admin user registry URLs, as listed in Step 2 in the `admin_user_registry_urls` section. Note that you must include all of the user registry URLs that you want to remain, listing each URL on a new line.
- `new_ldap_user_registry_url` is the URL of your new LDAP user registry, as determined in Step 1.

5. If you want to make your LDAP registry available for authenticating API Manager users, enter the following data, followed by a new line:

```
provider_user_registry_urls:
- >-
 current_provider_user_registry_urls
- new_ldap_user_registry_url
```

where:

- `current_provider_user_registry_urls` are the current provider organization user registry URLs, as listed in Step 2 in the `provider_user_registry_urls` section. Note that you must include all of the user registry URLs that you want to remain, listing each URL on a new line.
- `new_ldap_user_registry_url` is the URL of your new LDAP user registry, as determined in Step 1.

6. Press **CTRL D** to terminate the input. A confirmation message is output, for example:

```
user-registry-setting https://server.com/api/cloud/settings/user-registries
```

where `server.com` is your management server endpoint.

## Configuring your LDAP registry in a Catalog

If you want to make your LDAP registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. Complete the following steps:

1. Determine the URL of your LDAP user registry by using the following command (or you can copy and paste from the summary of the registry creation):

```
apic user-registries:list --server mgmt_endpoint_url --org admin
```

2. Log in to the management server as a member of a provider organization by entering the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

For full details of the `apic login` command, see [Logging in to a management server](#).

3. Enter the following command (the terminating hyphen character means that the command takes input from the command line):

```
apic configured-catalog-user-registries:create --server mgmt_endpoint_url --org organization_name --catalog catalog_name -
```

where `catalog_name` is the value of the `name` property of the required Catalog. The command returns:

```
Reading CONFIGURED_CATALOG_USER_REGISTRY_FILE arg from stdin
```

4. Enter the following data, followed by a new line:

```
user_registry_url: ldap_registry_url
```

where `ldap_registry_url` is the URL of your LDAP registry, obtained in step 1.

5. Press **CTRL D** to terminate the input.

## Switching your LDAP registry between writable and read-only

After an LDAP user registry has been created, it can be switched between writable and read-only by updating the `user_managed` property in the registry configuration. Complete the following steps.

1. Determine the name or ID of the LDAP user registry that you want to update, by running the following command (or you can use the summary from the registry creation):

```
apic user-registries:list --server mgmt_endpoint_url --org admin
```

The command returns a list of all the user registries on that server, shown by name followed by their registry URL. The registry ID is located at the end of the URL, for example `https://company.com/api/user-registries/x-x-x-x-x/registry_id`.

2. Enter the following command (the terminating hyphen character means that the command takes input from the command line):

```
apic user-registries:update --server mgmt_endpoint_url --org admin registry_name_or_id -
```

where `registry_name_or_id` is the name or ID of the LDAP user registry that you want to update (as determined in the previous step). The command returns:

```
Reading USER_REGISTRY_FILE arg from stdin
```

3. Enter the following data, followed by a new line:

```
user_managed: true_or_false
```

where `true` makes the registry writable, and `false` makes the registry read-only.

4. Press **CTRL D** to terminate the input.

Note that if you are changing your registry from read-only to writable, you must also set the `attribute_mapping` configuration, as described in the previous registry property tables.

## Using an LDAP user registry to secure APIs

If you want to use the LDAP user registry to secure APIs, see the following information:

- To use for basic authentication in the security definition for an API, see [Creating a basic authentication security definition](#).
- To use for authentication in the User Security configuration for a native OAuth provider, see [Configuring user security for a native OAuth provider](#).

For details of all the `apic user-registries` and `apic configured-catalog-user-registries` commands, see [apic user-registries](#) and [apic configured-catalog-user-registries](#).

## Example

The following example shows a configuration file that uses the Search DN authentication method for setting up writable LDAP:

```
name: sdn-ldap
title: "SDN LDAP User Registry"
integration_url: https://mycompany.com/api/cloud/integrations/user-registry/xxx-xxx-xxx
user_managed: true
user_registry_managed: false

case_sensitive: false
identity_providers:
- name: ldap
 title: "SDN LDAP Identity Provider"
endpoint:
 endpoint: "ldap://mycompany.com:389"
configuration:
 authentication_method: search_dn
 authenticated_bind: "true"
 admin_dn: "cn=admin,dc=company,dc=com"
 admin_password: xxxx
 search_dn_base: "dc=company,dc=com"
 search_dn_scope: sub
 search_dn_filter_prefix: (uid=
 search_dn_filter_suffix:)
 attribute_mapping:
 dn: "uid=[username],ou=users,dc=company,dc=com"
 cn: "[first_name] [last_name]"
 sn: "[last_name]"
 mail: "[email]"
 userPassword: "[password]"
```

## Related information

- [Command-line tool reference for the developer toolkit](#)
- [Securing your API Connect Cloud with LDAP \(series of developer articles\)](#)

## Configuring a Local User Registry

A Local User Registry (LUR) can be configured to provide user authentication for both Cloud Manager and API Manager.

### About this task

Local User Registries (LURs) are the default user registries included in API Connect. LURs are local databases included with API Connect. Two default LURs are installed and configured during installation of API Connect. They cannot be deleted. The default Admin user account is stored in the Provider LUR.


You can create and configure a new LUR. After you create it, you must set it as active in Settings > User Registries. See [Selecting user registries for Cloud Manager and API Manager](#).

One of the following roles is required to configure user registries:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage permissions`

### Procedure

Follow these steps to configure a new LUR:

1. In the Cloud Manager, click .
2. Click Create in the User Registries section.  
Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.
3. Select Local User Registry as the type for the user registry and enter the following information:

Field	Description
Title (required)	Enter a descriptive name for use on the screen.
Name (required)	The name that is used in CLI commands. The name is auto-generated. For details of the CLI commands for managing user registries, see <a href="#">apic user-registries</a> .
Display Name (required)	The name that is displayed for selection by the user when logging in to a user interface, or activating their API Manager account. For details of user interface log in, and account activation, see <a href="#">Accessing the Cloud Manager user interface</a> , <a href="#">Accessing the API Manager user interface</a> , and <a href="#">Activating your API Manager user account</a> . Note: The Developer Portal uses the <b>Title</b> of the User Registries when rendering them at the login page, rather than the <b>Display Name</b> .
Summary (optional)	Enter a brief description.
Case sensitive	Select this setting if user names are case-sensitive. Note: The Developer Portal does not support case sensitive usernames. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
Email required	Select this checkbox if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this checkbox if email addresses must be unique within the user registry. For new Local User Registries, this setting is always selected; so if email addresses are contained in the user record, they must be unique. However, for existing Local User Registries this setting can be edited. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

4. Click Save.

### Results

The user registry is saved and is available to be configured for user authentication in Cloud Manager or API Manager by selecting it in Settings > User Registries. See [Selecting user registries for Cloud Manager and API Manager](#).

### Related concepts

- [Managing authentication and security](#)
- [User registries overview](#)

### Related tasks

- [Configuring an Authentication URL user registry](#)
- [Configuring an LDAP user registry in the Cloud Manager](#)

- [Setting visibility for a user registry](#)

## Configuring an OIDC user registry

Configure a shared OIDC user registry for user onboarding and authentication when multi-factor authentication (MFA) is required.

You can create an OIDC user registry that is specific to a provider organization, or that is shared and available to all the provider organizations in your API Connect environment. An organization-specific OIDC user registry is used for onboarding and authenticating Developer Portal users, while a shared OIDC user registry can be used for onboarding and authenticating Cloud Manager, API Manager, and Developer Portal users.

This topic describes how to create a shared registry. For information on how to create an organization-specific registry, see [Creating an OIDC user registry](#).


Important: If you are using Twitter as your OIDC provider, API Connect supports only the OAuth 1.0a method, so your Twitter application must be configured to use OAuth 1.0a. Other OIDC providers support OAuth 2.0.

API Connect provides two methods for creating an OIDC user registry in Cloud Manager, as described in the following sections:

- [Using the UI to configure an OIDC user registry](#)
- [Using the CLI to configure an OIDC user registry](#)

## Using the UI to configure an OIDC user registry

Use the Cloud Manager application's user interface to configure a shared OIDC user registry when multi-factor authentication (MFA) is required.

1. In the Cloud Manager navigation pane, click  Resources.
2. Click User Registries.
3. Click Create and select OIDC User Registry.

Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.

4. On the Create OIDC User Registry page, use the fields in each of the following sections to configure the registry settings, and then click Create. Many of the registry settings are preconfigured to simplify the configuration steps.

### Provider Type

Use the settings in Table 1 to define the provider type.

Table 1. Provider Type settings

Field	Description
Provider Type	OIDC provider. Select one of the following supported OIDC providers: <ul style="list-style-type: none"> <li>• Facebook</li> <li>• GitHub</li> <li>• Google</li> <li>• LinkedIn</li> <li>• Slack</li> <li>• Twitter</li> <li>• Windows Live</li> <li>• Standard OIDC (default value allows you to specify another provider)</li> </ul>
Title	Provide a descriptive name for display purposes.
Name	Automatically generated. This name is used in CLI commands to reference the registry. For details of the CLI commands for managing user registries, see the <a href="#">apic user-registries</a> topic in the Command Line tool reference section of this documentation.
Summary	Provide a brief description of the new registry.
Email required	Select this check box if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this check box if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

### Provider Endpoint

Automatically generated for most supported providers. In the Authorization Endpoint field, type the URL of the provider's authorization endpoint.

### Token Endpoint

Fill in the settings as described in Table 2.

Table 2. Token Endpoint settings

Field	Description
URL	Preconfigured for most of the supported OIDC providers. Type the URL of the provider's token endpoint.
TLS	Select the TLS Client Profile for the token endpoint (OIDC must be configured to use TLS). Default is Default TLS Client Profile.

### UserInfo Endpoint

Fill in the settings as described in Table 3.

Table 3. UserInfo Endpoint settings

Field	Description
URL	Preconfigured for most of the supported OIDC providers. Type the URL of the provider's userinfo endpoint.
TLS	Select the TLS Client Profile for the userinfo endpoint (OIDC must be configured to use TLS). Default is Default TLS Client Profile.

### JWKS Endpoint

Fill in the settings as described in Table 4.

Table 4. JWKS Endpoint settings

Field	Description
URL	Type the URL of the read-only endpoint that contains the public keys' information in JWKS format.
TLS	Select the TLS Client Profile for the userinfo endpoint (OIDC must be configured to use TLS). Default is Default TLS Client Profile.

## Logout

Fill in the settings as described in Table 5.

Table 5. Logout

Field	Description
Logout redirect	Optionally, give a logout redirect URL, allowing APIC to initiate an https 302 redirect to the configured endpoint. Note: When the Developer Portal uses this feature, it does not retain control afterwards. Instead, the content is presented from the redirected endpoint.

## Client Information

Fill in the settings as described in Table 6.

Table 6. Client Information settings

Field	Description
Client ID	Provide the client ID of the application that is registered with the selected OIDC provider.
Client Secret	Provide the client secret of the application that is registered with the selected OIDC provider.
Response Type	Preconfigured for most of the supported OIDC providers. Specify the data type of the response that will be received from the OIDC provider.
Scopes	Preconfigured for most of the supported OIDC providers. Specify the access scope for the OIDC provider.
Client Authentication Method	Preconfigured for most of the supported OIDC providers. Select the authentication method to be used with the OIDC provider. Options are: <ul style="list-style-type: none"> <li>HTTP Basic authentication schema</li> <li>Data encoded form body</li> </ul>

## Additional Support

Optional. Select the additional security parameters described in Table 7.

Table 7. Additional security options

Security parameter	Setting name for CLI	Description
NONCE	<code>nonce</code>	Enable the NONCE extension to prevent compromised requests from being used again (replayed).
Proof Key for Code Exchange (PKCE)	<code>pkce</code>	Enable the PKCE extension to allow public clients to mitigate the threat of having the authorization code intercepted.

## Advanced Features

Optional. Select the advanced features described in Table 8.

Table 8. Advanced features

Feature / UI Label	Setting name for CLI	Description
Auto onboard	<code>auto_onboard</code>	Allow users to execute calls to APIs without logging in first, provided they present a valid token issued by the OIDC provider.
Always use the userinfo endpoint	<code>userinfo</code>	Configures the OIDC user registry to always fetch user data from the userinfo endpoint, if populated.
Use Portal as Endpoint for external OIDC provider traffic	<code>proxy_redirect</code>	When authenticating users, redirect the external OIDC provider to communicate with the Developer Portal instead of API Manager.
Return third-party access token	<code>proxy_access_token</code>	Include the third-party OIDC access token in the response. Note: Enable this setting for debugging purposes only. This setting is not recommended for use in a production environment. When this setting is enabled, the token size will increase when a request is made to API Connect using the token. The larger token size might exceed the HTTP protocol limit, resulting in an <code>ERR_HTTP2_PROTOCOL_ERROR</code> or <code>ERR_CONNECTION_CLOSED</code> error.
Return third-party id_token	<code>proxy_id_token</code>	Include the third-party OIDC id_token in the response. Note: Enable this setting for debugging purposes only. This setting is not recommended for use in a production environment. When this setting is enabled, the token size will increase when a request is made to API Connect using the token. The larger token size might exceed the HTTP protocol limit, resulting in an <code>ERR_HTTP2_PROTOCOL_ERROR</code> or <code>ERR_CONNECTION_CLOSED</code> error.
Token Relay	<code>token_relay</code>	Allow <code>access_token/refresh_token</code> to send in 302 redirect for logout
Userinfo POST	<code>post_userinfo</code>	If supported by your OIDC provider, by using HTTP POST method when contacting userinfo endpoint. Note: Not all OIDC providers support POST. Ensure that your OIDC provider supports this before you enable the feature.
Use IBM APIC token expiration setting from cloud	<code>override_provider_ttl</code>	Override the OIDC provider's token expiration setting with the access token and refresh token expiration settings that are configured in API Connect. For information on configuring the access token and refresh token expiration settings in API Connect, see <a href="#">Configuring timeouts for access tokens and refresh tokens</a> .
Disable hash verification (CLI only)	<code>disable_hash_verification</code>	Disable <code>at_hash, c_hash</code>

## User Mapping

Fill in the settings as described in Table 9.

Note:

The User Mapping fields are preconfigured for most of the supported OIDC providers to minimize potential errors; use care when changing the settings. For the Standard OIDC option, contact your OIDC provider to obtain the details of the fields.

Table 9. User mapping settings


Field	Description
Username	The name of the field in the response token that contains the user's user name. Note: The username field must be unique for this OIDC registry, because it identifies the user in the system and cannot be changed.
Email	The name of the field in the response token that contains the user's email address.
First name	The name of the field in the response token that contains the user's given name.
Last name	The name of the field in the response token that contains the user's surname.

#### Redirect URI

The Redirect URI section lists the endpoints to which the OIDC authorization server will redirect the authorization code. There is one endpoint for each API Connect organization type: Cloud administration, Provider organization, and Consumer organization. The redirect endpoint is required when a user registers their application with the OIDC provider. For example, if this OIDC user registry is used by a provider organization, the user must register the provider organization's redirect endpoint with the OIDC provider. These read-only fields are provided for you to copy the endpoint values as required.

### Enabling the OIDC user registry

Complete the following steps to enable the new user registry for Cloud Manager, API Designer, or both.

1. On the navigation pane, click  Settings.
2. Click User Registries, Edit.
3. Click Edit in the section that corresponds to the application for which you are enabling the new user registry.
4. Select the new OIDC user registry.
5. When you have finished enabling the registry, click Save.

## Using the CLI to configure an OIDC user registry

Use the developer toolkit CLI to configure a shared OIDC user registry when multi-factor authentication (MFA) is required.

You configure an OIDC user registry by first defining the registry details in a configuration file. You then use a CLI command to create the registry, passing the configuration file as parameter. To make the registry available to the Developer Portal, you must enable the registry in the associated Catalog.

Note:

- An OIDC registry, in common with a Local User Registry, cannot be used to secure APIs on the gateway.
- Because the interaction with the third party OIDC provider is handled by the Management server, the Management server is the application from the point of view of the third party OIDC provider. Your OIDC redirection endpoint, which is used by authorization server to send the token to the Management server, must be accessible to the OIDC provider through your firewall. When you register your application with the third party OIDC provider, you are required to supply the associated OIDC redirect URI, <https://consumer.mycompany.com/consumer-api/oauth2/redirect> for example. However, this information is not available until you have created your OIDC user registry in API Connect. You must therefore first register your application without this information, then update it later, as detailed in the instructions on this page.

### Logging in to the Management server

Before you can create an OIDC user registry, you must log in to your management server from the CLI. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm admin/identity_provider
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`.

For full details of the `apic login` command, see [Logging in to a management server](#).

### Defining your OIDC registry configuration

You define the configuration of your OIDC user registry in a YAML file. At a minimum, the YAML file must include the settings from the following list. For additional settings, see the tables provided with the UI version of the procedure in this topic.

```
title: registry_title
integration_url: oidc_integration_url
case_sensitive: case_sensitivity_setting
email_required: true_or_false
email_unique_if_exist: true_or_false
configuration:
 client_id: 'app_client_id'
 client_secret: 'my-client-secret'
 provider_type: oidc_provider_type
```

where:

- `registry_title` is your chosen descriptive title for the user registry.

- `oidc_integration_url` is the OIDC integration URL in your API Connect configuration. You can determine the OIDC integration URL by using the following CLI command:

```
apic integrations:list --server mgmt_endpoint_url --subcollection user-registry
```

- `case_sensitivity_setting` determines whether your user registry is case-sensitive. Valid values are:

- `true`
- `false`

To ensure proper handling of user name capitalization, you **must** ensure that your case-sensitivity setting here matches the setting on the backend OIDC provider:

- **Only** set `case_sensitive` to `true` if the backend OIDC provider supports case-sensitivity.
- Set `case_sensitive` to `false` if the backend OIDC provider does not support case-sensitivity.

Note: After at least one user has been onboarded into the registry, you cannot change this setting.

- `email_required` determines whether an email address is required as part of the user onboarding process. Valid values are:

- `true`
- `false`

If set to `true`, the source identity provider must supply the email address as part of the authentication process during onboarding.

Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.

- `email_unique_if_exist` determines whether email addresses must be unique within the user registry. Valid values are:

- `true`
- `false`

Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

- `app_client_id` is the client ID of the application that is registered with the OIDC server, and must be in string format.
- `my-client-secret` is the client secret of the application that is registered with the OIDC server, and must be in string format.
- `oidc_provider_type` is the type of OIDC provider; specify one of the following values:

- `facebook`
- `github`
- `google`
- `linkedin`
- `slack`
- `twitter`
- `windows_live`
- `standard`

Use the `standard` provider type for any OIDC provider that is compliant with the OIDC standard.

Note: If the provider type is `standard`, you must include the following additional properties in the `configuration` section of your YAML file:

```
authorization_endpoint: 'oidc_auth_endpoint'
token_endpoint:
 endpoint: 'oidc_token_endpoint'
```

where:

- `oidc_auth_endpoint` is the authorization endpoint on the OIDC server, and must be in string format.
- `oidc_token_endpoint` is the token endpoint on the OIDC server, and must be in string format.

## Default OIDC configurations

For each OIDC provider type, API Connect assumes a default configuration, but you can override the default configuration properties in your YAML file. The default configurations are as follows:

- Facebook

```
authorization_endpoint: 'https://www.facebook.com/v3.1/dialog/oauth'
token_endpoint:
 endpoint: 'https://graph.facebook.com/v3.1/oauth/access_token'
userinfo_endpoint:
 endpoint: 'https://graph.facebook.com/me'
scope: email public_profile
field_mapping:
 username: email
 email: email
 last_name: last_name
 first_name: first_name
```

- Github

```
authorization_endpoint: 'https://github.com/login/oauth/authorize'
token_endpoint:
 endpoint: 'https://github.com/login/oauth/access_token'
userinfo_endpoint:
 endpoint: 'https://api.github.com/user'
scope: 'read:user user:email'
field_mapping:
 username: login
 email: email
 last_name: name
 first_name: name
```

- Google

```
authorization_endpoint: 'https://accounts.google.com/o/oauth2/v2/auth'
token_endpoint:
 endpoint: 'https://www.googleapis.com/oauth2/v4/token'
scope: openid profile email
field_mapping:
 username: email
 email: email
 last_name: family_name
 first_name: given_name
```

- LinkedIn

```
authorization_endpoint: 'https://www.linkedin.com/oauth/v2/authorization'
token_endpoint:
 endpoint: 'https://www.linkedin.com/oauth/v2/accessToken'
userinfo_endpoint:
 endpoint: 'https://api.linkedin.com/v1/people/~:(id,first-name,last-name,picture-url,public-profile-url,email-address)?
format=json'
scope: r_basicprofile r_emailaddress
field_mapping:
 username: emailAddress
 email: emailAddress
 last_name: lastName
 first_name: firstName
credential_location: form_body
```

- Slack

```
authorization_endpoint: 'https://slack.com/oauth/authorize'
token_endpoint:
 endpoint: 'https://slack.com/api/oauth.access'
userinfo_endpoint:
 endpoint: 'https://slack.com/api/users.identity'
scope: identity.basic identity.email
field_mapping:
 username: user.email
 email: user.email
 last_name: user.name
 first_name: user.name
```

- Twitter

```
request_endpoint: https://api.twitter.com/oauth/request_token'
authorization_endpoint: https://api.twitter.com/oauth/authenticate'
token_endpoint:
 endpoint: 'https://api.twitter.com/oauth/access_token'
userinfo_endpoint:
 endpoint: 'https://api.twitter.com/1.1/account/verify_credentials.json'
oauth_signature_method: 'HMAC-SHA1'
field_mapping:
 email: email
 first_name: name
 last_name: name
 username: screen_name
```

- Windows Live

```
authorization_endpoint: 'https://login.microsoftonline.com/common/oauth2/v2.0/authorize'
token_endpoint:
 endpoint: 'https://login.microsoftonline.com/common/oauth2/v2.0/token'
scope: openid offline_access profile email
field_mapping:
 email: email
 username: preferred_username
 first_name: name
 last_name: name
```

- Standard

```
response_type: code
scope: openid
field_mapping:
 username: sub
 email: email
 last_name: family_name
 first_name: given_name
credential_location: auth_header
```

Although this is the default configuration for the **standard** provider type, you should contact your OIDC provider to obtain the details of the fields that you need to define.

## Creating your OIDC user registry

To create your OIDC user registry, use the following CLI command:

```
apic user-registries:create --server mgmt_endpoint_url --org admin oidc_config_file
```

where:

- *mgmt\_endpoint\_url* is the platform API endpoint URL.
- *organization\_name* is the value of the **name** property of your provider organization.
- *oidc\_config\_file* is the name of the YAML file that defines the configuration of your OIDC user registry.

On completion of the registry creation, the command displays the following summary details:

```
registry_name registry_url
```

By default, the *registry\_name* is derived from the **title** property in the configuration YAML file but you can override this by including a **name** property in the file. The *registry\_url* is an internal URL that API Connect assigns to the registry.

After you have created your OIDC user registry, you must update your application registration with the third party OIDC provider to include the OIDC redirect URI; you can obtain this information by using the following command, which displays the details of the registry in the command window:

```
apic user-registries:get --server mgmt_endpoint_url --org organization_name registry_name --output -
```



The required `oidc_redirect_uri` value is in the `consumer:` section; for example:

```
consumer:
 oidc_redirect_uri: https://consumer.mycompany.com/consumer-api/oauth2/redirect
```

## Enabling your OIDC registry in a Catalog

If you want to make your OIDC registry available for onboarding and authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. Complete the following steps:

1. Determine the URL of your OIDC user registry by using the following command:

```
apic user-registries:list --server mgmt_endpoint_url --org admin
```

2. Log in to the Management server as a member of a provider organization; enter the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

For full details of the `apic login` command, see [Logging in to a management server](#).

3. Enter the following command (the terminating hyphen character means that the command takes input from the command line):

```
apic configured-catalog-user-registries:create --server mgmt_endpoint_url --org organization_name --catalog catalog_name -
```

where `catalog_name` is the value of the `name` property of the required Catalog. The command returns

```
Reading CONFIGURED_CATALOG_USER_REGISTRY_FILE arg from stdin
```

4. Enter the following data, followed by a new line:

```
user_registry_url: oidc_registry_url
```

where `oidc_registry_url` is the URL of your OIDC registry, obtained in step [1](#).

5. Press **CTRL D** to terminate the input.

For details of all the `apic user-registries` and `apic configured-catalog-user-registries` commands, see [apic user-registries](#) and [apic configured-catalog-user-registries](#).

You can also complete the operations described in this topic by using the API Connect REST APIs; see the [API Connect REST API documentation](#).

---

## Setting visibility for a user registry

The visibility setting determines which provider organizations can access a resource, for example, user registries.

### Before you begin

---

You must complete the following tasks:

- [Creating an availability zone](#)
- [Registering a gateway service](#)

### About this task


---

The user registry visibility setting controls which provider organizations are able to use that registry for authenticating Developer Portal users, or for securing APIs. For details on how to specify the user registries that can be used for authenticating users of the Cloud Manager and API Manager user interfaces, see [Selecting user registries for Cloud Manager and API Manager](#).

### Procedure

---

Follow these steps to set the visibility for the resources (user registries) in your on-premises cloud:

1. In the Cloud Manager, click  Resources.
2. Select User Registries as the resource to view.
3. From the list of user registries, choose Edit visibility from the actions menu next to the name of the user registry you are working with.
4. Select the visibility setting for the user registry. The options are:
  - Private - the user registry is not visible and cannot be used by any provider organization
  - Public - the user registry is visible and can be used by all provider organizations
  - Custom - the user registry is visible only to the provider organizations designated by you
5. For Custom visibility, select the provider organizations that you want to have access to the user registry.
6. Click Make visible to complete the operation.

### Results

---

For Private, the user registry cannot be used by any provider organizations. For Public, the user registry can be used by all provider organizations. For Custom, the user registry can be used by the provider organizations that you designate.

---

## Deleting a user registry

As the Cloud Manager administrator, you can delete user registries.


## About this task

Complete the following steps to delete a user registry. All users within the user registry will be deleted. The pre-configured user registries (API Manager Local User Registry and Cloud Manager Local User Registry) may not be deleted. Any user registry marked as the Default in Settings, > User Registries also may not be deleted.

One of the following roles is required to delete a user registry:

- Administrator
- Owner

## Procedure

1. In the Cloud Manager, click .
2. Select User Registries to see the list of current user registries in your cloud.
3. Choose Delete from the overflow menu that is adjacent to the user registry you want to delete.
4. Optionally, you can delete multiple user registries by marking the checkboxes and selecting Delete Selected User Registries from the overflow menu for the list.
5. Confirm the deletion.

## Results

The user registry is deleted and removed from the User Registries list in Cloud Manager. All user accounts contained in the user registry are deleted.

## Removing a user from a user registry

As an API Connect user, you can remove yourself from an API Connect user registry. As an administrative user, you can remove other users from an API Connect user registry.

For details on how to remove a user from a user registry, see the following subtopics:

- [Removing yourself from a user registry](#)  
As an API Connect user, you can remove yourself from the API Connect user registry in which you were registered, by using the developer toolkit CLI.
- [Removing another user from a user registry](#)  
You can remove a user from an API Connect user registry by using the developer toolkit CLI.

## Removing yourself from a user registry

As an API Connect user, you can remove yourself from the API Connect user registry in which you were registered, by using the developer toolkit CLI.

## Before you begin

You cannot remove yourself from a user registry if you are a member of a provider organization. Before removing yourself from a user registry, ensure that you have been deleted from any provider organizations. For more information, see [Removing a user from an organization](#).

## Procedure

1. Log in to the management server as the user that you want to remove.  
The format of the login command depends on whether you are a user in the Cloud Manager admin organization, or you are a user in a provider organization.

If you are a Cloud Manager admin user, enter the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm admin/identity_provider
```

If you are a provider organization user, enter the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

If you want to log in as a member of the cloud administration organization, or as a member of a provider organization, you can help determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope scope --server mgmt_endpoint_url --fields title,realm
```

where *scope* has the value `admin` or `provider` depending on whether you want to log in as a member of the cloud administration organization, or as a member of a provider organization. The output lists the names and titles of all identity providers, for example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`, and the default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`. For full details of the `apic login` command, see [Logging in to a management server](#).

Note: If the same user registry is used for both the Cloud Manager and API Manager user interfaces, and you have access to both, when you remove yourself from the user registry you will lose access to both user interfaces regardless of which organization you log in to.

2. Remove yourself from the user registry. Enter the following command:

```
apic me:delete --server mgmt_endpoint_url
```

For example:

```
apic me:delete --server platform-api.myserver.com.com
```

The command confirms successful removal by returning the details of the deleted user; for example:

```
user1 [state: enabled] https://platform-api.myserver.com.com/api/user-registries/32830897-1d23-4fac-acf5-0193d0b2c1b5/4438937a-6ad0-4eaa-9163-820888ac6245/users/040adb11-e9a4-4d93-9c2e-62a974da0689
```

---

## Removing another user from a user registry

You can remove a user from an API Connect user registry by using the developer toolkit CLI.

### Before you begin

To complete this task, you must have Administrator access to your API Connect cloud, or be the owner of a provider organization, depending on the scope of the user registry.

You cannot remove a user from a user registry if that user is a member of the Cloud Manager admin organization, a provider organization, or a consumer organization, and that user registry is used for the associated login. Before removing a user from a user registry, ensure that they have been deleted from any organizations for which that user registry is used for login. For more information, see [Deleting a member](#) or [Removing a user from an organization](#).

### Procedure

1. Log in to the management server.
  - If the user registry is defined at the cloud administration level, and therefore available to all provider organizations, you must log in as a member of the Cloud Manager admin organization.
  - If the user registry is defined at a provider organization level, and therefore specific to that provider organization, you must log in as the organization owner.

For full details of the `apic login` command, see [Logging in to a management server](#).

Note: If the same user registry is used for both the Cloud Manager and API Manager user interfaces, and the user has access to both, when you remove them from the user registry they will lose access to both even though you are logging in to the admin organization.

2. Identify the name of the user registry from which you want to remove the user. Enter the following command:

```
apic user-registries:list --server mgmt_endpoint_url --org organization_name
```

where `organization_name` has the value `admin` if you are logged in as a member of the Cloud Manager admin organization, or is the name of the provider organization if you are logged in as an organization owner.

For example:

```
apic user-registries:list --server platform-api.myserver.com.com --org admin
```

The command returns a list of all user registries, with the registry name displayed first; for example:

```
api-manager-lur https://platform-api.myserver.com.com/api/user-registries/3283e897-1d23-4fac-acf5-0193d0b2c1b5/4438937a-6ad0-4eaa-9163-820888ac6245
cloud-manager-lur https://platform-api.myserver.com.com/api/user-registries/3283e897-1d23-4fac-acf5-0193d0b2c1b5/3adbf524-cd74-4051-99c0-89ce5ffcc9c0
my-ldap https://platform-api.myserver.com.com/api/user-registries/3283e897-1d23-4fac-acf5-0193d0b2c1b5/29eae413-cd74-4051-99c0-89ce5ffcc9c0
```

3. Identify the name of the user, from the required user registry, that you want to remove. Enter the following command:

```
apic users:list --server mgmt_endpoint_url --org organization_name --user-registry user_registry_name
```

For example:

```
apic users:list --server platform-api.myserver.com.com --org admin --user-registry my-ldap
```

The command returns a list of all users in the user registry, with the user name displayed first; for example:

```
user1 [state: enabled] https://platform-api.myserver.com.com/api/user-registries/32830897-1d23-4fac-acf5-0193d0b2c1b5/4438937a-6ad0-4eaa-9163-820888ac6245/users/040adb11-e9a4-4d93-9c2e-62a974da0689
user2 [state: enabled] https://platform-api.myserver.com.com/api/user-registries/32830897-1d23-4fac-acf5-0193d0b2c1b5/4438937a-6ad0-4eaa-9163-820888ac6245/users/12f1aa82-d9f0-4670-99f1-7500d1fb6583
```

4. Remove the required user from the user registry. Enter the following command:

```
apic users:delete user_name --server mgmt_endpoint_url --org organization_name --user-registry user_registry_name
```

For example:

```
apic users:delete user1 --server platform-api.myserver.com.com --org admin --user-registry my-ldap
```

The command confirms successful removal by returning the details of the deleted user; for example:

```
user1 [state: enabled] https://platform-api.myserver.com.com/api/user-registries/32830897-1d23-4fac-acf5-0193d0b2c1b5/4438937a-6ad0-4aaa-9163-820888ac6245/users/040adb11-e9a4-4d93-9c2e-62a974da0689
```

---

## TLS profiles overview

API Connect supports TLS Profiles for securing data transmission over HTTPS.

---

### Introduction to TLS profiles

Important: API Connect includes several default TLS profiles to help you get started working with the application. The default profiles should not be used in a production environment. It is important to create your own profiles to ensure a secure network.

API Connect may need to transmit data across an untrusted network, for example, when accessing the Gateway, email server, or LDAP server. TLS provides secure network layer transportation of data between two parties.

There are two types of TLS Profiles: a TLS Server Profile and a TLS Client Profile. A TLS Server Profile is used by the Gateway to configure its endpoint for use during API execution. A TLS Client profile is used whenever the system needs to communicate with another endpoint over TLS.

The components of a TLS Profile are:

- TLS Protocol version indicates the versions of the Transport Layer Security Protocol required for the profile. TLS Protocol versions 1.0, 1.1, 1.2, and 1.3 are supported.
- Optional support for mutual authentication and renegotiation for Server Profiles.
- Optional support for weak server connections and Server Name Indication for Client Profiles.
- Cipher suites to secure HTTPs communication within the API Connect ecosystem.
- Keystores containing public and private key pairs.
- Truststores containing public keys for trusted third party services, such as Google, Facebook, or Verisign.
- [Viewing TLS Profiles, Keystores, and Truststores](#)  
The current list of TLS profiles, Keystores, and Truststores can be viewed on the main TLS screen.
- [Creating a TLS Server Profile](#)  
In Cloud Manager, you can configure the profile that is used on the gateway when it acts as a TLS server.
- [Creating a TLS Client Profile](#)  
In Cloud Manager, TLS profiles are configured as a Resource to provide secure transmission of data over HTTPs.
- [Creating a Keystore](#)  
Each keystore contain a matched pair of a public certificates and its private keys. These artifacts provide identity information during a TLS handshake.
- [Creating a Truststore](#)  
A truststore contains a list of certificates. The certificates are used to verify the peer during a TLS handshake.
- [Viewing certificate details and adding certificates to a keystore or truststore](#)  
You can view details for the certificates in an existing keystore or truststore and add additional certificates.
- [Generating a self-signed certificate using OpenSSL](#)  
OpenSSL is an open source implementation of the SSL and TLS protocols. It provides the transport layer security over the normal communications layer, allowing it to be intertwined with many network applications and services.
- [Generating a PKCS#12 file for Certificate Authority](#)  
PKCS#12 (P12) files define an archive file format for storing cryptographic objects as a single file. API Connect supports the P12 file format for uploading a keystore and truststore. The keystore should contain both a private and public key along with intermediate CA certificates.
- [Binding a TLS server profile to a gateway service](#)  
You can bind the SSL certificate of a TLS profile to a gateway service to establish an HTTPS binding. This enables you to access the service by using the HTTPS communication protocol.
- [Updating the PKCS#12 certificate for a TLS server profile](#)  
A server certificate bound to a gateway service can be invalidated if the host name in the digital certificate of the server does not match the URL specified by the client, or because it has expired. When this happens, you must update the TLS profile with a new CA certificate or PKCS#12 (P12) file.

---

### Related tasks

- [Creating a TLS Server Profile](#)
- [Creating a TLS Client Profile](#)
- [Creating a Keystore](#)
- [Creating a Truststore](#)

---

## Viewing TLS Profiles, Keystores, and Truststores

The current list of TLS profiles, Keystores, and Truststores can be viewed on the main TLS screen.

---

### Before you begin

Important: API Connect includes several default TLS profiles to help you get started working with the application. The default profiles should not be used in a production environment. It is important to create your own profiles to ensure a secure network.

One of the following roles is required to view TLS Server Profiles:

- Administrator
- Owner
- Topology Administrator

- Viewer
- Custom role with the `Settings: Manage` or `Settings: view` permissions

## About this task





The TLS list screen provides a list of TLS Profiles, Keystores and Truststores configured for your cloud. Users with view permissions can use this screen as a reference for the TLS Profiles.

Users with manage permissions can create new Profiles, Keystores, and Truststores; edit existing ones; and delete items that are no longer needed. Users can view the Keystores and Truststores assigned to the profiles.

Users can view and set the visibility for Client Profiles. For more information on visibility, see [Setting visibility for a TLS Client Profile](#)

## Procedure

Following are the tasks you can perform using the list screen as a base:

1. In the Cloud Manager, click  Resources.
2. Select TLS.
3. Complete your management task:
  - To create a new item, click Create next to the table for the type of item you want to create.
  - To edit an existing item, either click the Title of the item, or click the options menu  and then click Edit.
  - To set the visibility of a TLS Client Profile, click the options menu  for the profile, and then click Edit Visibility.
  - To delete an existing item, click the options menu  for the profile, and then click Delete.

## Related concepts

- [TLS profiles overview](#)

## Related tasks

- [Creating a TLS Server Profile](#)
- [Creating a TLS Client Profile](#)
- [Creating a Keystore](#)
- [Creating a Truststore](#)

## Creating a TLS Server Profile

In Cloud Manager, you can configure the profile that is used on the gateway when it acts as a TLS server.

## Before you begin

Important: API Connect includes several default TLS profiles to help you get started. The default profiles should not be used in a production environment. It is important to create your own profiles to secure your network.

One of the following roles is required to configure TLS Server Profiles:


- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: manage` permissions

## About this task

The Server profile is for the gateway when it is acting as the TLS server.

## Procedure

Perform the following steps to create a TLS Server profile:

1. In the Cloud Manager, click  Resources.
2. Select TLS.
3. Click Create in the TLS Server Profile table.
4. Enter the fields to configure the TLS Server Profile:

Field	Description
Title (required)	Enter a Title for the profile. The title is displayed on the screen.
Name (required)	The Name is auto-generated. The value in the Name field is a single string that can be used in developer toolkit CLI commands. To view the CLI commands to manage TLS Server Profiles, see <a href="#">apic tls-server-profiles</a> .
Version (required)	Assign a version number for the profile. Using version numbers allows you to create multiple server profiles with the same name and different configurations, for example, <i>MyProfile 1.0</i> and <i>MyProfile 1.1</i> .
Summary (optional)	Enter a description of the profile.

Field	Description
Protocols (required)	Select one or more supported TLS protocol versions. The default is 1.2.
Mutual Authentication (required)	Determines the level of two-way authentication for the server profile. In two-way authentication, the server responds to a client by sending a request for the client certificate. <ul style="list-style-type: none"> <li>None (default) No support for mutual authentication.</li> <li>Request Enable this option to request client authentication during the TLS handshake. When the application sends the request, the gateway requests that the application sends the certificate. If the client does not send the certificate, the certificate is not checked on the gateway.</li> <li>Require Enable this option to require client authentication during the TLS handshake. When the application sends the request, the gateway requests that the application sends the certificate. If the client does not send the certificate, the TLS handshake fails and the request is blocked.</li> </ul>
Limit Renegotiation (optional)	Client-initiated renegotiation allows the connection to be retried. The default is to prevent renegotiation. Remove the checkmark to allow renegotiation.
Keystore (required)	A keystore is a repository containing a public and private key pair. The Server Profile requires a keystore in order to securely identify the system. When an application sends an API request, the keystore is used to verify a matching certificate.
Truststore (optional)	A truststore is a repository containing certificates. The certificates are used to verify the peer during a TLS handshake. If, in addition to a keystore, a truststore is specified, the certificate is further checked for validity by ensuring that is signed by the root certificate, which must be in the truststore.
Ciphers (required)	Cipher suites are encryption/decryption algorithms used to secure HTTPs communication within the API Connect ecosystem. Select the ciphers that the profile supports. Note: The TLS 1.3 ciphers are clearly indicated. If you select TLS version 1.3 as one of the protocols for the profile but do not select any TLS 1.3 ciphers, <b>all</b> the TLS 1.3 ciphers are added to the list of ciphers supported by the profile. If you do not select TLS version 1.3 but select one or more TLS 1.3 ciphers, those ciphers are <b>not</b> added to the list of ciphers supported by the profile.

5. Click Save.

## Related concepts

- [TLS profiles overview](#)

## Related tasks

- [Creating a TLS Client Profile](#)
- [Creating a Keystore](#)
- [Creating a Truststore](#)

## Creating a TLS Client Profile

In Cloud Manager, TLS profiles are configured as a Resource to provide secure transmission of data over HTTPs.

### Before you begin

One of the following roles is required to configure TLS Profiles:


- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage permissions`

### About this task

API Connect uses both TLS Server and TLS Client profiles. A TLS Server profile is presented when a communication request is received. The Server profile validates the request against the Keystore and protocol version to determine whether the connection is secure. The Client profile is presented to initiate communication with another system. Client Profiles may be made visible for use by provider organizations in policies in API Manager. See [Setting visibility for a TLS Client Profile](#).

### Procedure

Perform the following steps to create a TLS Client profile:

1. In the Cloud Manager, click  Resources.
2. Select TLS.
3. Click Create in the TLS Client Profile table.
4. Enter the fields to configure the TLS Client Profile:

Field	Description
Title (required)	Enter a Title for the profile. The title is displayed on the screen.
Name (required)	The Name is auto-generated. The value in the Name field is a single string that can be used in developer toolkit CLI commands. To view the CLI commands to manage a TLS Client Profile, see <a href="#">apic tls-client-profiles</a> .
Version (required)	Assign a version number for the profile. Using version numbers allows you to create multiple server profiles with the same name and different configurations, for example, <i>MyProfile 1.0</i> and <i>MyProfile 1.1</i> .
Summary (optional)	Enter a description of the profile.

Field	Description
Protocols (required)	Select one or more supported TLS protocol versions. The default is 1.2.
Server Connection (optional)	Specify whether to support weak or insecure credentials. <ul style="list-style-type: none"> <li>Allow insecure server connections - Insecure server connections may result from self-signed certificates, expired or corrupted certificates, or certificates from an unknown or untrusted source. Check this box to allow the connection to proceed with an insecure connection. The default is to not allow insecure server connections.</li> <li>Support Server Name Indication (SNI) - Check this box to enable SNI. SNI allows support for multiple certificates presented on the same IP address using different host names. The client profile sends the name of a virtual domain as part of the TLS negotiation. The default is to enable SNI.</li> </ul>
Keystore (optional)	A Keystore is a repository containing public and private key pairs. Select the keystore where you will store the certificates for the profile. Default keystores are provided, and you can also create your own.
Truststore (optional)	A Truststore is a repository containing verified public keys, which are usually obtained from a third-party certificate authority. Truststores provide list of certificates to verify a peer's certificate. If used in a TLSProfile, a truststore is used when mutual authentication is enabled. Select a truststore for the profile. Default truststores are provided, and you can also create your own.
Ciphers (required)	Cipher suites are encryption/decryption algorithms used to secure HTTP communication within the API Connect ecosystem. Select the ciphers that the profile supports. Note: The TLS 1.3 ciphers are clearly indicated. If you select TLS version 1.3 as one of the protocols for the profile but do not select any TLS 1.3 ciphers, <b>all</b> the TLS 1.3 ciphers are added to the list of ciphers supported by the profile. If you do not select TLS version 1.3 but select one or more TLS 1.3 ciphers, those ciphers are <b>not</b> added to the list of ciphers supported by the profile.

5. Click Save.

- [Setting visibility for a TLS Client Profile](#)  
The visibility setting determines which provider organizations can access a resource.
- [Defining elliptic curve cryptographic schemes for a TLS client profile](#)  
You define the elliptic curve cryptographic schemes for a TLS client profile by using the developer toolkit CLI.

## Related concepts

- [TLS profiles overview](#)

## Related tasks

- [Setting visibility for a TLS Client Profile](#)
- [Creating a TLS Server Profile](#)
- [Creating a Keystore](#)
- [Creating a Truststore](#)

## Setting visibility for a TLS Client Profile

The visibility setting determines which provider organizations can access a resource.

### Before you begin

You must complete the following task:



- [Creating a TLS Client Profile](#)

### About this task

The visibility setting gives you control over the resources that are available to provider organizations.

## Procedure

Follow these steps to set the visibility for the resources in your on-premises cloud:

1. In the Cloud Manager, click  Resources.
2. Select TLS as the resource to view.
3. From the table of TLS Client Profiles, choose Edit visibility from the  options menu next to the name of the profile you are managing.
4. Select the visibility setting for the profile. The options are:
  - Private - the profile is visible only in Cloud Manager and cannot be used by any provider organization
  - Public - the profile is visible and can be used by all provider organizations
  - Custom - the profile is visible only to the provider organizations designated by you
5. For Custom visibility, select the provider organizations that you want to have access to the profile.
6. Click Save to complete the operation.

## Results

For Private, the profile cannot be used by any provider organizations. For Public, the profile can be used by all provider organizations. For Custom, the profile can be used by the provider organizations that you designate. If visible, the profiles will be available in API Manager.

# Defining elliptic curve cryptographic schemes for a TLS client profile

You define the elliptic curve cryptographic schemes for a TLS client profile by using the developer toolkit CLI.

## About this task

To define elliptic curve cryptographic schemes for a TLS client profile, you include `elliptic_curve_auto_negotiation` and `elliptic_curve` properties in a YAML file definition for the TLS client profile. The `elliptic_curve` property lists the required elliptic curve cryptographic schemes. For example:

```
elliptic_curve_auto_negotiation: false
elliptic_curve:
 - secp521r1
 - secp384r1
 - prime256v1
```

You then use the developer toolkit CLI to create the TLS client profile in API Connect.

When `elliptic_curve_auto_negotiation` is set to `true`, the system negotiates the Elliptic-curve Diffie-Hellman (ECDH) key agreement automatically with its peer, and any `elliptic_curve` property settings are ignored.

The following example shows a complete YAML file for a TLS client profile:

```
type: tls_client_profile
name: my-tls-client-profile
version: 1.0.0
title: My TLS client profile
protocols:
 - tls_v1.2
ciphers:
 - ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
elliptic_curve_auto_negotiation: false
elliptic_curve:
 - sect163k1
insecure_server_connections: false
server_name_indication: true
```

Note:

- The `elliptic_curve_auto_negotiation` option is not supported by any of the API Connect gateway types. If the TLS client profile is targeted for an API Connect gateway; this setting is ignored by the gateway.
- The elliptic curve cryptographic schemes shown in each row of the following table are equivalent. However, API Connect recognizes only one or the other depending on how the TLS profile is used, as indicated in the table.

Table 1.

API enforcement on the gateway	API Connect server access security
secp192r1	prime192v1
secp256r1	prime256v1

Therefore, if you want to use either of these schemes and are unsure whether you are targeting the TLS client profile to the API Connect gateway for API enforcement, whether you are using it to secure user access to the API Connect servers, or whether it will be used for both purposes, specify both equivalent schemes; API Connect will simply ignore the non-relevant scheme. For example:

```
elliptic_curve:
 .
 .
 .
 - secp256r1
 - prime256v1
 .
 .
 .
```

## Procedure

To create a TLS client profile with elliptic curve cryptographic schemes defined, complete the following steps:

- Create a YAML file definition for your TLS client profile, with the required `elliptic_curve` property.
- Log in to the management server from the developer toolkit CLI. Log in either as a member of the cloud administration organization or as a member of a provider organization, depending on where you want to create the TLS client profile. For details, see [Logging in to a management server](#).
- Create the TLS client profile by using the following command:

```
apic tls-client-profiles:create --server mgmt_endpoint_url --org organization_name tls_client_profile_yaml_file
```

where:

- `mgmt_endpoint_url` is the platform API endpoint URL, and is the same as that which was used when you logged in at step 2.
- `organization_name` is either `admin`, for the cloud administration organization, or the name of your provider organization, and is the same as that which was used when you logged in at step 2.
- `tls_client_profile_yaml_file` is the name of the YAML file that contains the definition for your TLS client profile.

Note: When you install IBM® API Connect, the API Connect gateway has a pre-supplied default TLS client profile that is used for API enforcement if you do not configure a TLS client profile; you cannot configure this default TLS client profile on the gateway.

For reference details of all the `apic tls-client-profiles` commands, see [apic tls-client-profiles](#).

You can also complete the operations described in this topic by using the API Connect REST APIs; see the [API Connect REST API documentation](#).



---

## Creating a Keystore

Each keystore contain a matched pair of a public certificates and its private keys. These artifacts provide identity information during a TLS handshake.

### Before you begin

---

Cloud Manager and API Manager both support and use TLS certificates, but they do not themselves produce strong encryption keys or manage your encryption keys. Encryption keys are generated and managed according to your own procedures. For more information, see [Generating a PKCS#12 file for Certificate Authority](#) and [Generating a self-signed certificate using OpenSSL](#).

One of the following roles is required to configure Keystores:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage permissions`

### About this task


---

API Connect includes pre-configured Keystores which may be used for testing purposes. For production environments, we suggest creating a new, secure Keystore.

### Procedure

---

Perform the following steps to create a TLS Client profile:

1. In the Cloud Manager, click  Resources.
2. Select TLS.
3. Click Create in the Keystore table.

Field	Description
Title (required)	Enter a Title for the Keystore. The title is displayed on the screen.
Name (required)	The Name is auto-generated. The value in the Name field is a single string that can be used in developer toolkit CLI commands. To view the CLI commands to manage keystores, see <a href="#">apic keystores</a> .
Summary (optional)	Enter a brief description.
Private Key & Public Key: Step 1: Upload private key	Upload the file containing the private key certificate. If necessary, you can click Browse to locate the file. If the file contains both the private and public keys, upload it in Step 1. Private and public keys are always uploaded in pairs, either in a single file or separate files.
Private key password (optional)	Enter the password for the private key if it has a password.
Private Key & Public Key: Step 2: Upload public key	If the public key is contained in a separate file, upload it in Step 2. Private and Public keys are always uploaded in pairs, either in a single file or separate files.

4. Click Save.

Note: After they have been uploaded, private keys cannot be downloaded from API Connect.

### Related concepts

---

- [TLS profiles overview](#)

### Related tasks

---

- [Creating a TLS Client Profile](#)
- [Creating a Truststore](#)
- [Generating a PKCS#12 file for Certificate Authority](#)
- [Generating a self-signed certificate using OpenSSL](#)

---

## Creating a Truststore

A truststore contains a list of certificates. The certificates are used to verify the peer during a TLS handshake.

### Before you begin

---

One of the following roles is required to configure Truststores:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage permissions`

### About this task


---

Cloud Manager and API Manager both support and use TLS certificates, but they do not themselves produce strong encryption keys or manage your encryption keys. Encryption keys are generated and managed according to your own procedures. For more information, see [Generating a PKCS#12 file for Certificate Authority](#) and [Generating a self-signed certificate using OpenSSL](#).

API Connect includes pre-configured Truststores which may be used for testing purposes. For production environments, we suggest creating a new, secure Truststore.

## Procedure

---

1. In the Cloud Manager, click  Resources.
2. Select TLS.
3. Click Create in the Truststore table.

Field	
Title (required)	Enter a Title for the Truststore. The title is displayed on the screen.
Name (required)	The Name is auto-generated. The value in the Name field is a single string that can be used in developer toolkit CLI commands. To view the CLI commands to manage truststores, see <a href="#">apic truststores</a> .
Summary (optional)	Enter a brief description.
Public Keys	Upload the file containing the public key certificate. If necessary you can click Browse to locate the file.

4. Click Save.

## Related concepts

---

- [TLS profiles overview](#)

## Related tasks

---

- [Creating a TLS Client Profile](#)
- [Creating a Keystore](#)

# Viewing certificate details and adding certificates to a keystore or truststore

---

You can view details for the certificates in an existing keystore or truststore and add additional certificates.

## Before you begin

---

One of the following roles is required to edit Keystores and Truststores:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage permissions`

## About this task


---

API Connect includes pre-configured Keystores and Truststores which you can use for testing purposes. For production environments, we suggest creating a new, secure Keystore or Truststore. From an existing Keystore or Truststore, you can edit the title, view the details of the certificates, and add new certificates.

## Procedure

---

To work with existing keystores and truststores:

1. In the Cloud Manager, click  Resources.
2. Select TLS.
3. Click the name of the keystore or truststore.
4. Edit the name and summary if needed.
5. The Subject, Finger Print, and Expiration date is shown for each certificate in the keystore or truststore. Click > to view the certificate details.
6. Add additional private and/or public keys if needed.
7. Click Save.

Note: After they have been uploaded, private keys cannot be downloaded from API Connect.

## Related concepts

---

- [TLS profiles overview](#)

## Related tasks

---

- [Creating a TLS Server Profile](#)
- [Creating a TLS Client Profile](#)
- [Creating a Truststore](#)
- [Creating a Keystore](#)
- [Generating a PKCS#12 file for Certificate Authority](#)
- [Generating a self-signed certificate using OpenSSL](#)

---

# Generating a self-signed certificate using OpenSSL

OpenSSL is an open source implementation of the SSL and TLS protocols. It provides the transport layer security over the normal communications layer, allowing it to be intertwined with many network applications and services.

---

## Before you begin

One of the following roles is required to complete this task:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage permissions`

---

## About this task

This topic tells you how to generate a self-signed SSL certificate request using the OpenSSL toolkit to enable HTTPS connections.

---

## Procedure

To generate a self-signed SSL certificate using the OpenSSL, complete the following steps:

1. Write down the Common Name (CN) for your SSL Certificate. The CN is the fully qualified name for the system that uses the certificate. For static DNS, use the hostname or IP address set in your Gateway Cluster (for example, `192.16.183.131` or `dp1.acme.com`).
2. Run the following OpenSSL command to generate your private key and public certificate. Answer the questions and enter the Common Name when prompted.

```
openssl req -newkey rsa:2048 -nodes -keyout key.pem -x509 -days 365 -out certificate.pem
```

3. Review the created certificate:

```
openssl x509 -text -noout -in certificate.pem
```


4. Combine your key and certificate in a PKCS#12 (P12) bundle:

```
openssl pkcs12 -inkey key.pem -in certificate.pem -export -out certificate.p12
```

5. Validate your P2 file.

```
openssl pkcs12 -in certificate.p12 -noout -info
```

Once the certificate file is created, it can be uploaded to a keystore.

6. In the Cloud Manager, click  Resources.
7. Select TLS.
8. Click Create in the Keystore table.
9. Create a Keystore and upload the certificate file following the instructions at [Creating a Keystore](#).

Note:

- API Connect supports only the P12 (PKCS12) format file for the present certificate.
- Your P12 file must contain the private key, the public certificate from the Certificate Authority, and all intermediate certificates used for signing.
- Your P12 file can contain a maximum of 10 intermediate certificates.

10. Click Save.

---

## Related concepts

- [TLS profiles overview](#)

---

## Related tasks

- [Creating a TLS Server Profile](#)
- [Creating a TLS Client Profile](#)
- [Creating a Keystore](#)
- [Creating a Truststore](#)

---

# Generating a PKCS#12 file for Certificate Authority

PKCS#12 (P12) files define an archive file format for storing cryptographic objects as a single file. API Connect supports the P12 file format for uploading a keystore and truststore. The keystore should contain both a private and public key along with intermediate CA certificates.

---

## Before you begin

One of the following roles is required to add a key to a keystore or truststore:

- Administrator
- Owner
- Topology Administrator

- Custom role with the `Settings: Manage permissions`

Before you can generate a P12 file, you must have a private key (for example: `key.pem`), a signed certificate by a Certificate Authority (for example `certificate.pem`) and one or more certificates from the CA authority.

Note: If your certificate file contains more than one certificate, you must manually split the file and create a single file for each entry. Each entry must be bound by the following markers:

```
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
```

## Procedure

1. If you have intermediate certificates from your CA, concatenate them into a single `.pem` file to build your `caChain`. Be sure to enter a new line following each certificate's data.


```
cat ca1.pem ca2.pem ca3.pem > caChain.pem
cat caChain.pem
-----BEGIN CERTIFICATE-----
MIIEPjCCA46gAwIBAgIQEod26KZabjd+BQMG1Dwl6jANBgkqhkiG9w0BAQUFADCBC
...
lQX7CkTJn61AJUsyEa8H/gjVQnHp4VOLFR/dKgeVcCRvZF7Tt5AuiyHY
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIEPDCCAySgAwIBAgIQSEus8arH1xND0aJ0NUmXJTANBgkqhkiG9w0BAQUFADBVB
...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIEENjCCAx6gAwIBAgIBATANBgkqhkiG9w0BAQUFADBvMQswcQYDVQGEwJTRTEU
...
-----END CERTIFICATE-----
```

2. Create the P12 file including the private key, the signed certificate and the CA file you created in step 1, if applicable. Omit the `-CAfile` option if you don't have CA certificates to include.

The following command uses OpenSSL, an open source implementation of the SSL and TLS protocols.

```
openssl pkcs12 -inkey key.pem -in certificate.pem -export -out certificate.p12 -CAfile caChain.pem -chain
```

Once the certificate file is created, it can be uploaded to a keystore.

3. In the Cloud Manager, click  Resources.
4. Select TLS.
5. Click Create in the Keystore table.
6. Create a Keystore and upload the certificate file following the instructions at [Creating a Keystore](#).
 

Note:

  - API Connect supports only the P12 (PKCS12) format file for the present certificate.
  - Your P12 file must contain the private key, the public certificate from the Certificate Authority, and all intermediate certificates used for signing.
  - Your P12 file can contain a maximum of 10 intermediate certificates.
7. Click Save.

## Related concepts

- [TLS profiles overview](#)

## Related tasks

- [Creating a TLS Server Profile](#)
- [Creating a TLS Client Profile](#)
- [Creating a Keystore](#)
- [Creating a Truststore](#)
- [Generating a self-signed certificate using OpenSSL](#)

## Binding a TLS server profile to a gateway service

You can bind the SSL certificate of a TLS profile to a gateway service to establish an HTTPS binding. This enables you to access the service by using the HTTPS communication protocol.

## Before you begin

One of the following roles is required:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Topology: Manage permissions`


For more information on which user roles have access, see [Adding members to the admin organization](#).

## About this task

Perform the following steps to bind a TLS profile to a gateway service.

## Procedure

---

1. In the Cloud Manager user interface, click  Topology, then select the gateway service that you want to work with.
2. Scroll down to the API Invocation Endpoint section, and under Server Name Indication (SNI) select the required TLS server profile. You can also bind a TLS server profile during the initial registration of a gateway service; see [Registering a gateway service](#).
3. Click Save when done.

## Results

---

The TLS profile binds to the gateway service.

## Related tasks

---

- [Creating a TLS Server Profile](#)

---

## Updating the PKCS#12 certificate for a TLS server profile

A server certificate bound to a gateway service can be invalidated if the host name in the digital certificate of the server does not match the URL specified by the client, or because it has expired. When this happens, you must update the TLS profile with a new CA certificate or PKCS#12 (P12) file.

## Before you begin

---

One of the following roles is required:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage permissions`

## About this task

---


If the expiration date of a certificate or a P12 file is approaching, or if a certificate is invalidated, use the steps in this topic to update a TLS profile bound to a gateway service. CA certificate and P12 file expiration dates are displayed in the details of the containing keystore; see step [3](#).

You update the certificate for a TLS server profile by replacing the certificate in the keystore that is associated with the TLS server profile.

Complete the following steps to update a TLS profile that has an invalidated or expired certificate or P12 file.

## Procedure

---

1. In the Cloud Manager user interface, click  Resources, then click TLS.
2. To identify the keystore that is associated with the TLS server profile, complete the following steps:
  - a. In the TLS Server Profile section, select the required profile.
  - b. In the Keystore/Truststore section, note the selected keystore, then click Cancel to close the TLS server profile details page.
3. In the Keystore section, select the required keystore. The Certificates section displays the expiration date of a certificate. You can expand a certificate to see further details.
4. Click Browse and select the required P12 file.

Note:

  - API Connect supports only the P12 (PKCS12) format file for the present certificate.
  - Your P12 file must contain the private key, the public certificate from the Certificate Authority, and all intermediate certificates used for signing.
  - Your P12 file can contain a maximum of 10 intermediate certificates.
5. Click Save when done.

## Related tasks

---

- [Creating a TLS Server Profile](#)

---

## OAuth Provider overview

API Connect supports OAuth Specification 2.0, for both Native and Third party implementations.

## Introduction to OAuth

---

OAuth is a token-based authorization protocol that allows third-party websites or applications to access user data without requiring the user to share personal information. In API Connect, you can secure an API with OAuth.

In Cloud Manager, you configure both Native and Third party OAuth providers that can be made visible to selected Provider organizations. The OAuth Provider configuration is based on the OAuth 2.0 Specification, which is available at <https://tools.ietf.org/html/rfc6749>. Knowledge of the OAuth 2.0 specification is required to implement an OAuth Provider in API Connect.

One of the following roles is required to configure OAuth Providers:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

Note: In a multi-node cluster, OAuth operations will fail if quorum is lost. Quorum requires that the number of active nodes is greater than 50% of the total number of nodes in the cluster.

- [Configuring a native OAuth provider](#)  
Native OAuth providers are configured and managed by you within your cloud.
- [Configuring a third-party OAuth provider](#)  
Enter the secure endpoints to provide OAuth authentication from a third party.
- [Setting the visibility for OAuth providers](#)  
The visibility setting determines which provider organizations can use an OAuth provider to secure APIs.
- [OAuth concepts for API Connect](#)

---


## Configuring a native OAuth provider

Native OAuth providers are configured and managed by you within your cloud.

### About this task

---

A native OAuth provider object provides settings for OAuth processing operations such as generating and validating OAuth tokens. An OAuth provider object is referenced by an OAuth security definition to protect an API. When a native OAuth provider is used, the OAuth operations are performed natively by API Connect.

Every OAuth provider object has a backing API. Your configuration here automatically updates the OpenAPI document of the API. You can edit the OpenAPI document directly by navigating to the  Resources > OAuth Providers page, selecting your OAuth provider, then clicking API Editor.

Note: Take care when modifying the code directly on the Source tab of the API Editor because validation is limited. For example:

- If you change the name of auto generated assembly actions in the source code, the assembly will be prevented from updating dynamically when the OAuth provider settings are modified.
- You must ensure that the OAuth provider name matches the value specified in the `oauth-provider-settings-ref` field in each OAuth assembly action.

When a published API references an OAuth provider object, the backing API is automatically made available in the gateway.

Token requests and `client_id` checks:

If you are calling an OAuth endpoint, the sequence of `client_id` checks in the token request is as follows:

1. Check both body and query.
  - If found in only the body, validate and return a **200** or appropriate return code.
  - If found in only the query, return a **wrong location** error.
  - If found in both the body and the query, return a **more than one location** error.
2. When not found in the body or the query, check the **Authorization** header.
  - If found in this header, validate and return a **200** or appropriate return code.
3. Not found, return the appropriate error and code.

The best location for the `client_id` is in the request body.

One of the following roles is required to configure a native OAuth provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions


Note:

- The OAuth provider logs Analytics data for failure cases, but does not log successful cases. Activity log policies that call for logging of Analytics data upon success do not apply for the OAuth provider.
- You must ensure the OAuth Provider is configured in the Sandbox Catalog before using the OAuth Provider in a non-Sandbox Catalog.

## Procedure

---

OAuth provider configuration uses a series of screens. The first set of screens controls a basic OAuth provider configuration. Perform the following steps to configure a native OAuth Provider for your cloud:

1. In the Cloud Manager, click  Resources.
2. Click OAuth Providers > Add > Native OAuth provider.
  - a. Complete the following parameters for the first screen, then click Next.

Field	Description
Title	Enter a title for the native OAuth provider.
Name	This field is auto-populated by the system.

Field	Description
Description (optional)	Enter a brief description.
Base path (optional)	The base path is the URL segment of the API that is shared by all operations in the API. It does not include the host name or any additional segments for paths or operations. The base path must be unique for a given catalog. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
Gateway Type	Select the gateway type, either DataPower® Gateway (v5 compatible) or DataPower API Gateway. For information about types of gateways, see <a href="#">API Connect gateway types</a> .  OAuth Providers apply to one gateway type.

b. In the next screen, enter the following additional configuration parameters, then click Next.

Field	Description
Authorize Path	/oauth2/authorize/ is the standard OAuth endpoint to login to account
Token Path	/oauth2/token/ is the standard OAuth endpoint to exchange code for access token.
Supported grant types	<ul style="list-style-type: none"> <li>• Implicit - An access token is returned immediately without an extra authorization code exchange step.</li> <li>• Application - Application to application. Corresponds to the OAuth grant type "Client Credentials." Does not require User Security.</li> <li>• Access code - An authorization code is extracted from a URL and exchanged for an access code. Corresponds to the OAuth grant type "Authorization Code."</li> <li>• Resource owner - Password - The user's username and password are exchanged directly for an access token, so can only be used by first-party clients.</li> <li>• <span style="border: 1px solid green; padding: 2px;">API Gateway only</span> Resource owner - JWT - A verified signed JSON Web Token is exchanged directly for an access token.</li> </ul>
Supported client types	<ul style="list-style-type: none"> <li>• Confidential - Client can maintain secure credentials on a secure server</li> <li>• Public - Client credentials are not secure.</li> </ul>

c. Enter the scopes in the next screen. A scope becomes an option in the request and response for an access token. Click Add to add additional fields for scopes. Click Next when done.

Field	Description
sample_scope_1	Scope for token
additional scopes	Scope for token

d. Enter the parameters for User Security in the next screen. Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization. User Security is not required for the Application grant type. Click Next when done.

Field	Description
Identity Extraction	<p>Determines how the user credential is extracted:</p> <ul style="list-style-type: none"> <li>• Basic Authentication - HTTP basic authentication (requires no additional configuration)</li> <li>• Default HTML Form - Use default login form for user name and password</li> <li>• <span style="border: 1px solid green; padding: 2px;">API Gateway only</span> Context variable - Specify which variable contains the user name and password. API Connect OAuth context variables as listed here <a href="#">API Connect context variables</a> Note: DataPower API Gateway only. Not available for DataPower Gateway (v5 compatible).</li> <li>• Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form. For instructions on creating a custom form, see <a href="#">Creating a custom HTML login form for user security</a>.</li> <li>• Redirect - Enter an endpoint to redirect to a third-party identity provider. For more information, see <a href="#">Authenticating and authorizing through a redirect URL</a>.</li> <li>• <span style="border: 1px solid green; padding: 2px;">API Gateway only</span> Disabled - do not collect the user credential</li> </ul> <p>Note: If you use either the Default HTML Form or Redirect identity extraction methods, the response from the redirect endpoint <b>must</b> maintain the order of the query parameters before the <b>state_nonce</b> query parameter, otherwise the authorization fails.</p>
Authentication	Authenticate application users with a user registry. Select an LDAP or Authentication URL user registry or create the SampleAuthURL User Registry. For a DataPower API Gateway, you have the option to disable authentication with a user registry.
Authorization	<p>The following methods for extracting the user credential are available:</p> <ul style="list-style-type: none"> <li>• Authenticated - Authorize authenticated users automatically.</li> <li>• Default HTML Form - Use default HTML form to authorize.</li> <li>• Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form. For instructions on creating a custom form, see <a href="#">Creating a custom HTML authorization form for user security</a>.</li> <li>• <span style="border: 1px solid green; padding: 2px;">API Gateway only</span> Disabled - Disable authorization.</li> </ul>

3. Review the Summary for the native OAuth provider configuration.

4. Click Back to make changes.

5. Click Finish to save the basic configurations and to proceed to the Advanced Parameters for a native OAuth Provider.

## Results

Depending upon the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

- [Configuring basic settings for a native OAuth provider](#)  
You can update the identification details and basic configuration settings for a native OAuth provider.
- [Configuring scopes for a native OAuth provider](#)  
Access tokens contain authorization for specific scopes.
- [Configuring user security for a native OAuth provider](#)  
Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization.
- [Configuring tokens for a native OAuth provider](#)  
Set time to live for access tokens and refresh tokens, and a time period for maximum consent for all tokens.
- [Configuring token management and revocation for a native OAuth provider](#)  
Select whether to use a native gateway (DataPower) or third party endpoint for token revocation.
- [Configuring introspection for a native OAuth provider](#)  
Define an introspection path to allow the metadata for an access token to be examined.

- [Configuring metadata for a native OAuth provider](#)  
Use Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.
- [Configuring the OIDC parameters for a native OAuth provider](#)  
Open ID Connect (OIDC) provides an additional authentication protocol based on OAuth 2.0. OIDC provides user information encoded in a JSON Web Token, or JWT.
- [Editing a native OAuth provider by using the API Editor](#)  
You can edit the source and assembly policies for the Native OAuth Provider using the API editor.

## Configuring basic settings for a native OAuth provider


You can update the identification details and basic configuration settings for a native OAuth provider.

### About this task

One of the following roles is required to configure the basic settings for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

You can select the basis settings pages for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the basic settings for an existing native OAuth provider. If you want to update the basic settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click  Resources, OAuth Providers.
2. Select the required native OAuth provider.

### Procedure

1. To modify the identification details, click Info in the sidebar menu, then update the following fields as required:

Field	Description
Title	Enter a title for the native OAuth provider.
Name	This field is auto-populated by the system.
Description (optional)	Enter a brief description.
Base path (optional)	The base path is the URL segment of the API that is shared by all operations in the API. It does not include the host name or any additional segments for paths or operations. The base path must be unique for a given catalog. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.

2. To modify the basic configuration settings, click Configuration in the sidebar menu, then update the following fields as required:

Field	Description
Authorize Path	<code>/oauth2/authorize/</code> is the standard OAuth endpoint to login to account
Token Path	<code>/oauth2/token/</code> is the standard OAuth endpoint to exchange code for access token.
Supported grant types	<ul style="list-style-type: none"> <li>• Implicit - An access token is returned immediately without an extra authorization code exchange step.</li> <li>• Application - Application to application. Corresponds to the OAuth grant type "Client Credentials." Does not require User Security.</li> <li>• Access code - An authorization code is extracted from a URL and exchanged for an access code. Corresponds to the OAuth grant type "Authorization Code."</li> <li>• Resource owner - Password - The user's username and password are exchanged directly for an access token, so can only be used by first-party clients.</li> <li>• <span style="border: 1px solid green; padding: 2px;">API Gateway only</span> Resource owner - JWT - A verified signed JSON Web Token is exchanged directly for an access token.</li> </ul> <p>Tip: Selecting only the Resource owner - JWT grant type when defining a native OAuth provider in the user interface is not supported and results in an invalid configuration. To avoid this problem, additionally select either the Access code or the Resource owner - Password grant type.</p> <p>Note: If you plan to configure OpenID Connect (OIDC) for a native OAuth provider, you must include at least one of the following grant types: Implicit, Access code.</p>
Supported client types	<ul style="list-style-type: none"> <li>• Confidential - Client can maintain secure credentials on a secure server</li> <li>• Public - Client credentials are not secure.</li> </ul>

DataPower Gateway (v5 compatible) only Note: If the gateway type is DataPower® Gateway (v5 compatible) and, when the native OAuth provider was created, only the Application grant type was selected, you cannot add further grant types until you configure the user security settings. In particular, you must specify the user registry for authenticating application users. To configure the user security settings, complete the following steps:

- a. Click User Security in the sidebar menu, then click Edit.
  - b. Update the user security settings as required; for more details, see [Configuring user security for a native OAuth provider](#).
  - c. Click Save when done.
3. Click Save when done.

## Configuring scopes for a native OAuth provider

Access tokens contain authorization for specific scopes.



## About this task

Client applications can request only the scopes or a subset of the scopes that you define here. The scopes are included in the access tokens that are generated from the provider. When an OAuth protected API is invoked, the gateway checks the scopes carried in the access tokens against the list of allowed scopes in the security definition for the API to determine whether to grant access.


In addition, you can enforce advanced scope checks. The advanced scope check URLs are invoked after application authentication or after user authentication based on which URLs are configured. The final scope permission that is granted by the access token is the result of all scope checks.

Per IETF RFC 6749, the value of the scope parameter is a list of space-delimited, case-sensitive strings. For more information, see [The OAuth 2.0 Authorization Framework](#).

One of the following roles is required to configure scopes for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

You can select the scope settings page for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the scope settings for an existing native OAuth provider. If you want to update the scope settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click  Resources > OAuth Providers.
2. Select the required native OAuth provider.

## Procedure

Perform the following steps to configure the scopes for the access token:

1. Click Scopes in the sidebar menu. The currently configured scopes are listed. Review and update the scopes as required.

Field	Description
sample_scope_1	Scope for token
additional scopes	Scope for token

In the Default scopes section, select the default scopes to be used if the API request doesn't contain any scopes. If no default scope is defined, and the request doesn't contain a scope, an invalid scope error is returned for the request.

If the user authorization method is set to Default HTML Form in the User Security settings, all scopes specified here are added automatically to the authorization consent form.

2. Advanced scope check before token generation. This setting specifies the scope check endpoint where additional scope verification is performed in addition to the basic scopes. The advanced scope check URLs are invoked after application authentication or after owner authentication based on which URLs are configured. The scopes are included in the token and will overwrite any previous scopes.

Field	Description
Application scope check	Allow extra verification by running a scope check from an endpoint. Enter the endpoint and an optional TLS Profile to use for an application scope check.
Owner scope check	Further refine the scope with an additional check. Enter the endpoint and an optional TLS Profile to use for an owner scope check.

For more information about scope, see [Scope](#)

3. Advanced scope check after token generation. This setting specifies an additional scope check at the API consumer level to verify compliance with the scope requirements of the API.

Field	Description
Enabled	Select the check box to enable the advanced scope check after token validation. Enter an optional default validator endpoint.
Use endpoint from API	Select the check box to use the endpoint from the API, or clear the check box to override the endpoint from the API.

For more information about scope, see [Scope](#)

4. Click Save when done.

## Results

Depending upon the visibility setting, the OAuth Provider with the specified scopes can be used to secure the APIs in catalog.

## Related information

- [Scope](#)

## Configuring user security for a native OAuth provider

Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization.


## About this task

User security authenticates the user. It is required for the Implicit, Access code, and Resource owner - Password grant types. It is not used for the Application or Resource owner - JWT grant types.

One of the following roles is required to configure user security for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions




You can select the user security settings page for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the user security settings for an existing native OAuth provider. If you want to update the user security settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click  Resources, OAuth Providers.
2. Select the required native OAuth provider.

## Procedure

Perform the following steps to configure the user security settings for the OAuth Provider:

1. Click User Security in the sidebar menu.
2. Specify the following parameters for User Security. Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization. User Security is not required for the Application or Resource owner - JWT grant types. Click Next when done.

Field	Description
Identity Extraction	<p>Determines how the user credential is extracted:</p> <ul style="list-style-type: none"> <li>• Basic Authentication - HTTP basic authentication (requires no additional configuration)</li> <li>• Default HTML Form - Use default login form for user name and password</li> <li>•  Context variable - Specify which variable contains the user name and password. API Connect OAuth context variables as listed here <a href="#">API Connect context variables</a></li> <li>• Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form. For instructions on creating a custom form, see <a href="#">Creating a custom HTML login form for user security</a>.</li> <li>• Redirect - Enter an endpoint to redirect to a third-party identity provider. For more information, see <a href="#">Authenticating and authorizing through a redirect URL</a>.</li> <li>•  Disabled - do not collect the user credential</li> </ul> <p>Note: If you use either the Default HTML Form or Redirect identity extraction methods, the response from the redirect endpoint <b>must</b> maintain the order of the query parameters before the <code>state_nonce</code> query parameter, otherwise the authorization fails.</p>
Authentication	<p>Authenticate application users with a user registry. Select an LDAP or Authentication URL user registry or create the SampleAuthURL User Registry.</p>
Authorization	<p>Various methods may be used to authorize application users. For a DataPower® API Gateway, the following methods for extracting the user credential are available:</p> <ul style="list-style-type: none"> <li>• Authenticated - Authorize authenticated users automatically.</li> <li>• Default HTML Form - Use default HTML form to authorize. If you select the Default HTML Form method, all scopes that are specified in the Scopes settings are added automatically to the authorization consent form.</li> <li>• Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form.</li> <li>•  Disabled - Disable authorization.</li> </ul>

3. Click Save when done.

## Results

Depending upon the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

## Related information

- [Creating a custom HTML login form for user security](#)
- [Creating a custom HTML authorization form for user security](#)

## Configuring tokens for a native OAuth provider

Set time to live for access tokens and refresh tokens, and a time period for maximum consent for all tokens.

## About this task


Access tokens are granted to the client application to allow the application to access resources on behalf of the application user. Refresh tokens are issued to the client to obtain a new access token when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or more narrow scope. You can also specify how long the consent given by the combination of any number of access and refresh token remains valid.

One of the following roles is required to configure tokens for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

You can select the token settings page for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the token settings for an existing native OAuth provider. If you want to update the token settings for an existing native OAuth provider, complete the




following steps before following the procedure described in this topic:

1. Click  Resources, OAuth Providers.
2. Select the required native OAuth provider.

## Procedure

Perform the following steps to configure tokens for the native OAuth provider:

1. Click Tokens in the sidebar menu.
2. Define the settings to configure tokens.

Field	Description
Access tokens time to live	Enter the expiration time period in seconds for access tokens.
 One time use access token	Click the check box to enable one time use for the access token. Access tokens are multiple use by default which allows them to be used for multiple requests. When one time use is enabled, the access token will be consumed after one use. The OAuth flow will need to be repeated to obtain another access token. Note: If you select this option, you must also enable token management; see one of the following topics, depending on the user interface you are using: <ul style="list-style-type: none"><li>• <a href="#">Configuring token management and revocation for a native OAuth provider</a> (Cloud Manager)</li><li>• <a href="#">Configuring token management and revocation for a native OAuth provider</a> (API Manager)</li></ul>
Refresh tokens	Click the check box to enable Refresh tokens. Set the Count to limit the number of times a refresh token can be issued. Set the Refresh Token Time to Live value to determine the time to live, or expiration time period, for each refresh token in seconds.
One time use refresh token	Clear the check box to disable one time use for the refresh tokens. Refresh tokens are one time use by default which allows them to be used one time only to generate an access token and a new refresh token. When refresh token one time use is disabled then the refresh token count is limited to one and the refresh token can be used multiple times to generate new access tokens, however, another refresh token will not be generated unless the initial OAuth flow (Authorization Code or Password) is repeated. Note: If you select this option, you must also enable token management; see one of the following topics, depending on the user interface you are using: <ul style="list-style-type: none"><li>• <a href="#">Configuring token management and revocation for a native OAuth provider</a> (Cloud Manager)</li><li>• <a href="#">Configuring token management and revocation for a native OAuth provider</a> (API Manager)</li></ul>
Maximum consent	Click the check box to enable Maximum consent and enter the Maximum Consent Time to Live value in seconds. This is the time to live, or expiration time period, for all tokens, both access and refresh.
 Token secret	Click the check box to select the Shared Secret which was configured for the gateway. If no Shared Secret was entered in the Gateway Configuration, then enter a key name and key value to use as the token secret.
 Proof Key for Code Exchange	Proof Key for Code Exchange (PKCE) is a method to protect OAuth 2.0 public clients from an authorization code interception attack when they use Authorization Code grant requests. You can enable this extension when deploying with the DataPower® API Gateway. For more information, see <a href="#">RFC 7636</a> . Select the options for your OAuth Providers: <ul style="list-style-type: none"><li>• Enable proof key for code exchange If selected, enforces PKCE when submitted in Authorization Code grant requests.</li><li>• Always required If selected, requires PKCE in all Authorization Code grant requests.</li><li>• Allow plain Select this check box to allow the plain challenge method in Authorization Code grant requests.</li></ul>

3. Click Save when done.

## Results

Depending upon the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

## Related information

- [Refresh tokens](#)

## Configuring token management and revocation for a native OAuth provider

Select whether to use a native gateway (DataPower) or third party endpoint for token revocation.

## About this task

Token management enables you to prevent replay attacks by configuring token revocation. API Connect supports token revocation using a native gateway (DataPower) or a third party endpoint. For a native gateway, quota enforcement is used to manage tokens. For a third party endpoint, a URL to an external service is used to manage tokens.


For more information, see the IETF RFC 7009 [OAuth 2.0 Token Revocation](#).

One of the following roles is required to configure token management and revocation for a native OAuth Provider:

- Administrator
- Owner

- Topology Administrator
- Custom role with the `Settings:Manage` permissions

You can select the token management settings page for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the token management settings for an existing native OAuth provider. If you want to update the token management settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click  Resources > OAuth Providers.
2. Select the required native OAuth provider.

## Procedure

Perform the following steps to configure revocation settings for tokens:

1. Select Token Management in the sidebar menu.
2. Enable token management by selecting the check box.
3. From the Type list, select either Native or External. Native points to DataPower as the token storage location; External points to a revocation URL for token storage.  
Note: If you are using the API Manager user interface then, for the External option to be available, you must be using DataPower® API Gateway Version 10.0.1.0 or later, and the gateway service must be enabled in the Sandbox Catalog; for details on how to enable a gateway service in a Catalog, see [Creating and configuring Catalogs](#).
4. For Native, select one or both of the Resource owner revocation path and Client revocation path.
  - Resource owner revocation path - Uses the standard OAuth revocation path to allow the resource owner (end user) to revoke the application permission.
  - Client revocation path - Uses the standard OAuth revocation path to allow the client (application) to revoke a single token when the application closes.
 For more information about managing tokens with the Native DataPower Gateway, see [Token management with the native DataPower Gateway](#).
5. For External, the settings depend on the gateway type, as follows:
 

DataPower API Gateway:

  - Endpoint - the URL of the external management endpoint.
  - TLS Client Profile (optional) - the TLS client profile to secure connections.
  - Security - how to secure connections. The only supported method is basic authentication.
  - Basic authentication username (optional) - the user name for authentication.
  - Basic authentication password (optional) - the password for authentication.
  - Basic authentication request header name (optional) - the request header that contains the authentication string; if you supply both a request header name and user name/password, the request header authentication method is used.
  - Custom header pattern (optional) - the name pattern of the headers to use for sending additional information to the external management service.
  - Cache type - the cache type to control whether and how to cache positive responses. If you select Time to live, specify how long to keep responses in the cache; the default value is 900 seconds.
  - Fail on error - if selected, processing is stopped if the connection to the external management service fails.

Note: For details of the JSON format that is required when exchanging messages with the external management service, see [JSON format to exchange messages with the external management service](#).

DataPower Gateway (v5 compatible):

  - Endpoint - Enter the URL to an external web server that contains information about access or refresh tokens. API Connect calls the URL to determine if the associated token can be trusted. The token server then checks a token *blocklist* (a data store of inactive tokens) to ensure that the token is still valid. If the token is still valid, API Connect continues the processing. For more information see [Token revocation](#).
  - TLS Client Profile - Select a TLS profile to verify the external endpoint.
6. Click Save when done.

## Results

Depending on the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

## Configuring introspection for a native OAuth provider

Define an introspection path to allow the metadata for an access token to be examined.


### About this task

Token introspection allows an authorized holder of an access token to examine the contents of tokens using an introspection path. The access token to introspect must be obtained through the native OAuth provider. Introspection provides context for the token by allowing an authorized protected resource to query the authorization server to determine the set of metadata for a given token. The metadata includes whether or not the token is currently active, the scopes assigned to the token, and the authorization context in which the token was granted (including who authorized the token and which client it was issued to). API Connect token introspection conforms to IETF RFC 7662. See [OAuth 2.0 Token Introspection](#).

One of the following roles is required to enable introspection for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

You can select the introspection settings page for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the introspection settings for an existing native OAuth provider. If you want to update the introspection settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click  Resources > OAuth Providers.
2. Select the required native OAuth provider.

## Procedure

---

Perform the following steps to enable introspection:

1. Click Introspection in the sidebar menu.
2. Select the check box to enable Introspection. The OAuth standard path for introspection, /oauth2/introspect is automatically entered. This path will be used when another entity inspects the token contents.
3. Click Save when done.

## Results

---

Tokens will be queried using the /oauth2/introspect path. Depending upon the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

---

## Configuring metadata for a native OAuth provider

Use Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.

### About this task

---

Configure an Authentication URL or an External URL from which custom metadata is collected for inclusion in the token. The metadata is either stored inside the access token or it is sent along with the access token to the client application. For more information about how the metadata is collected, see [OAuth external URL and authentication URL](#).


Following are examples of metadata that can be included with the access token:

- Metadata about the authenticated resource owner
- Grant type that was used to obtain the token
- A confirmation code to be provided to the client application

One of the following roles is required to configure metadata collection for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions



You can select the metadata settings page for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the metadata settings for an existing native OAuth provider. If you want to update the metadata settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click  Resources, OAuth Providers.
2. Select the required native OAuth provider.

## Procedure

---

Perform the following steps to configure metadata collection:

1. Click Metadata in the sidebar menu.
2. Select Collect metadata to enable metadata collection.
3. The Authentication URL user registry is selected by default and is required. For more information about the Authentication URL, see [Authentication URL user registry](#).
4. Select the External URL to collect metadata from an external URL. Enter the endpoint and an option TLS Client Profile.
5.  If required, override the default Header name token value. The value of this header, if returned in the response from the OAuth endpoint, is placed in the response payload and indicated as **metadata**.
6.  If required, override the default Header name payload value. The value of this header, if returned in the response from the OAuth endpoint, is placed within the access token and indicated as **miscinfo**.
7. Save the OAuth Provider.
8. Click Save when done.

## Results

---

Depending upon the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

---

## Configuring the OIDC parameters for a native OAuth provider

Open ID Connect (OIDC) provides an additional authentication protocol based on OAuth 2.0. OIDC provides user information encoded in a JSON Web Token, or JWT.

### About this task

---


When you enable OpenID connect, a template is provided for generating ID tokens along with access tokens and the required assembly policies are automatically created. You can customize the policies to suit your needs in the API Editor. The sample key provided is for test purposes only and is used to sign the JWT token.

One of the following roles is required to configure an OIDC template for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

Note: You can configure OIDC parameters only if the selected grant types for the native OAuth provider include at least one of the Implicit or Access code grant types; see [Configuring basic settings for a native OAuth provider](#).

You can select the OIDC settings page for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the OIDC settings for an existing native OAuth provider. If you want to update the OIDC settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click  Resources, OAuth Providers.
2. Select the required native OAuth provider.

## Procedure

Perform the following steps to configure an OIDC template:

1. Click OpenID Connect in the sidebar menu.
2. Select the initial check box to configure an OIDC Template. Enter the following parameters:

Field	Description
<input checked="" type="checkbox"/> API Gateway only Support hybrid response types (optional)	Select the response types for the OpenID Connect hybrid flow to be supported by this OAuth provider.
<input checked="" type="checkbox"/> API Gateway only Auto Generate OIDC API Assembly	Select this option to generate the full OIDC assembly. Leave this option unselected to simply enable OIDC support in the OAuth provider and allow the developer to implement their own assembly.
<input checked="" type="checkbox"/> DataPower Gateway (v5 compatible) only ID token issuer	Descriptive text to indicate the source of the key.
<input checked="" type="checkbox"/> DataPower Gateway (v5 compatible) only ID token signing key	Specify the JSON Web Key (JWK) to be used to sign the ID token.
<input checked="" type="checkbox"/> DataPower Gateway (v5 compatible) only ID token signing algorithm	Select the algorithm used to sign the token.

3. Click Save when done. You can edit the policies by using the API Editor.

## Results

Depending upon the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

## Editing a native OAuth provider by using the API Editor

You can edit the source and assembly policies for the Native OAuth Provider using the API editor.

### About this task

If you have configured an OIDC template, you can customize it in API Editor. In the API Editor, the Source tab allows you to edit the code for the configuration using a text editor. The API Assemble tab provides a graphical drag-and-drop editor (identical to the one in API Manager) that allows you to add additional elements to the assembly for the OAuth Provider.


Note: Take care when modifying the code directly on the Source tab of the API Editor because validation is limited. For example:

- If you change the name of auto generated assembly actions in the source code, the assembly will be prevented from updating dynamically when the OAuth provider settings are modified.
- You must ensure that the OAuth provider name matches the value specified in the `oauth-provider-settings-ref` field in each OAuth assembly action.

One of the following roles is required to configure tokens for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

You can modify the native OAuth provider configuration by selecting the API Editor page immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the configuration for an existing native OAuth provider. If you want to update the configuration for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click  Resources, OAuth Providers.
2. Select the required native OAuth provider.

## Procedure

Perform the following steps to edit the OAuth configuration:

1. Click API Editor in the sidebar menu.
2. In the Source tab, view and edit the policies to customize the behavior for the OAuth provider.
3. In the API Assemble tab, use the drag and drop editor to add additional policies to the OIDC behavior.  
Note: If you add a policy that references a TLS profile, an **invoke** policy for example, then when you publish an API that uses this OAuth provider, you must ensure that the TLS profile is enabled for the Catalog to which you publish the API. For details on how to enable a TLS profile in a Catalog, see [Creating and configuring Catalogs](#).
4. Click Save when done.

## Results

Depending upon the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

## Configuring a third-party OAuth provider

Enter the secure endpoints to provide OAuth authentication from a third party.


### About this task


One of the following roles is required to configure OAuth Providers:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions


## Procedure

Complete the following steps to configure a third party OAuth provider:

1. In the Cloud Manager, click  Resources.
2. Select OAuth Providers > Add > Third party OAuth Provider.
  - a. Complete the following parameters for the first screen and click Next.

Field	Description
Title	Enter a descriptive title for the gateway service. This title will be displayed on the screen.
Name	This field is auto-populated by the system and used as the internal field name.
Supported grant types	Select from the following options: <ul style="list-style-type: none"><li>• Implicit: An access token is returned immediately without an extra authorization code exchange step.</li><li>• Application: Application to application. Corresponds to the OAuth grant type Client Credentials. Does not require User Security.</li><li>• Access code: An authorization code is extracted from a URL and exchanged for an access code. Corresponds to the OAuth grant type Authorization Code.</li><li>• Resource owner - Password: The user's username and password are exchanged directly for an access token, so can only be used by first-party clients.</li><li>•  Resource owner - JWT: a JSON Web Token (JWT) Bearer Token is used as a means for requesting an OAuth 2.0 access token, and for client authentication, as defined by the <a href="#">JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants</a>.</li></ul>
Gateway type	Select the gateway type, either DataPower® Gateway (v5 compatible) or DataPower API Gateway. For information about types of gateways, see <a href="#">API Connect gateway types</a> .  OAuth Providers apply to one gateway type.

- b. Specify configuration settings for the endpoints.

Field	Description
Authorization URL	An authorization URL where the resource owner grants authorization to the client application to access a protected resource. Example:  <code>https://example.com/oauth2/authorize</code>
Token URL	A token request URL where the client application exchanges an authorization grant for an access token. Example:  <code>https://example.com/oauth2/token</code>
Introspect URL	The introspection URL is where the API gateway validates the access tokens that are issued by the third party provider. Example:  <code>https://example.com/oauth2/introspect</code>  For more information on integrating third party OAuth providers for introspection, see <a href="#">OAuth introspection for third-party OAuth providers</a> .
 Introspect cache type	The cache type determines how long responses from the third party provider are cached, if at all. Select one of the following options: <ul style="list-style-type: none"><li>• No cache (default): Responses are not cached.</li><li>• Protocol: Defined by the <code>cache-control</code> header in the provider response.</li><li>• Time to live: Defined by the provider.</li></ul>

Field	Description
<input type="checkbox"/> API Gateway only Cache Time to Live	The length of time, in seconds, for which provider responses are cached, if the Introspect cache type is set to Time to live. The default value is 900.
TLS Profile (optional)	Select an optional TLS profile for communicating with the third party provider.
Security	Default is Basic Authentication.
Basic authentication request header name	The <code>x-introspect-basic-authorization-header</code> is available to provide a user-configured HTTP Basic authorization header.
<input type="checkbox"/> API Gateway only Basic authentication username (optional)	The default user name for HTTP Basic authentication.
<input type="checkbox"/> API Gateway only Basic authentication password (optional)	The default password for HTTP Basic authentication.
<input type="checkbox"/> API Gateway only Token validation	Specifies the method used to determine the success of the introspection request that is sent to the third party service to validate the provided token. Select one of the following options: <ul style="list-style-type: none"> <li>• Connected: The query is successful if the status return code is <b>200</b>.</li> <li>• Active (default): The query is successful if the status return code is <b>200</b> and the response JSON body includes the property <b>active: true</b>.</li> </ul>
<input type="checkbox"/> API Gateway only Custom header pattern (optional)	A regular expression for request headers that are to be passed to the third-party provider; for example, <code>x-Introspect-*</code> .
<input type="checkbox"/> API Gateway only Authorization header pass through	Select this check box if you want to retain the <b>Authorization</b> header for a bearer token. The default behavior is to remove this header.

c. Enter the scopes in the third screen. A scope becomes an option in the request and response for an access token. Click Add to add additional fields for scopes. Click Next when done.

Field	Description
sample_scope_1	Scope for token
sample_scope_2	Scope for token
additional scopes	Scope for token

d. Review the settings on the Summary panel.

3. Click Save and Edit to complete the configuration.

## Setting the visibility for OAuth providers

The visibility setting determines which provider organizations can use an OAuth provider to secure APIs.


### About this task

One of the following roles is required to configure OAuth Providers:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

### Procedure

Follow these steps to set the visibility for the OAuth providers in your on-premises cloud:

1. In the Cloud Manager, click  Resources.
2. Select OAuth Providers.
3. Choose Edit visibility from the actions menu next to the name of the OAuth provider that requires a visibility setting.
4. Select the visibility setting for the OAuth provider. The options are:
  - Private - the OAuth provider is not visible and cannot be used by any provider organization. It is accessible only to the admin organization in Cloud Manager.
  - Public - the OAuth provider is visible and can be used by all provider organizations
  - Custom - the OAuth provider is visible only to the provider organizations that you designate
5. For Custom visibility, select the provider organizations that you want to have access to the OAuth provider.
6. Click Save to complete the operation.

### Results

For Private, the OAuth provider is not visible and cannot be used by any provider organizations. For Public, the OAuth provider is visible and can be used by all provider organizations. For Custom, the OAuth provider is visible only to the provider organizations designated by you.

## OAuth concepts for API Connect

OAuth is a token-based authorization protocol that allows third-party websites or applications to access user data without requiring the user to share personal information.



Note: In a multi-node cluster, OAuth operations will fail if quorum is lost. Quorum requires that the number of active nodes is greater than 50% of the total number of nodes in the cluster.

- [OAuth user scenario](#)  
Potential users of OAuth with IBM® API Connect have a number of methods to secure their API. The following scenario provides an overview of the available options.
- [OAuth introspection for third-party OAuth providers](#)  
OAuth token validation can be offloaded to the third-party Open Authorization (OAuth)/Open ID Connect (OIDC) provider by using the Introspection URL. Clients can use a third-party OAuth or OIDC provider to obtain a token that is protected by IBM API Connect. IBM API Connect can use this feature along with the mentioned provider to protect access to the API.
- [OAuth external URL and authentication URL](#)  
You can use the Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.
- [Scope](#)  
Per IETF RFC 6749, the value of the scope parameter is expressed as a list of space-delimited, case-sensitive strings.
- [Tokens](#)  
Tokens can be managed using refresh tokens and revocation URLs. You can extend the life of tokens using refresh tokens. You can end the life of a token by specifying a revocation URL.
- [Authentication URL user registry](#)  
You can use an Authentication URL user registry to specify a REST authentication service that manages user authentication, and optionally provides additional metadata to be embedded in the token.
- [Custom forms for user security](#)  
You can create custom forms for the authorization and identity extraction phase of OAuth.
- [Securing an API with a JSON Web Token](#)  
There are two methods to secure your API with a JSON Web Token. You can use the **jwt-generate** command, or you can use a token that has been generated external to IBM API Connect.
- [Troubleshooting OAuth](#)  
You can find the answers to some common questions in the Developer Portal, or in support communities and forums in DeveloperWorks, on GitHub, or on YouTube.

---

## Related information

- [The OAuth 2.0 Framework](#)

---

## OAuth user scenario

Potential users of OAuth with IBM® API Connect have a number of methods to secure their API. The following scenario provides an overview of the available options.

---

### Scenario overview

In this scenario, Alice is a user of an application. Alice can grant permission for an application to access specific information about Alice in a third-party system that is using OAuth. Depending on the type of OAuth that is supported by the target service, Alice does not enter her user name and password into the application. Instead, the application receives an access token that represents her credentials (user name and password). The application can now access the information about Alice in the target system.

For example, Alice maintains a list of books that she reads on a service that is provided by mybooks.com. Following the purchase of a smartphone, Alice installs an application on her new phone to display the book details. The phone application wants to call an API provided by mybooks.com, which can access the information. The mybooks.com API is secured by using the OAuth protocol.

To access the book details, the application must complete a two-step process:

1. The application must first obtain permission from Alice.
2. The application then uses that permission to call the target service and obtain the list of books.

In the first step, the application typically directs Alice to the provider of the target service, mybooks.com. Alice provides her user name and password, and gives permission for the application to access her information. It is important that Alice trusts that she is providing her credentials to the provider of the target service and not to an untrusted proxy application. For example, by checking that the security certificate of the website where Alice enters her credentials matches what Alice expects from the provider of the target service.

The result of this step is the access token that the application can use to call the API. The application then generates the appropriately formatted OAuth request. For example, the Authorization header, or HTTP query parameters, which includes the access token, consumer key, and signature method that are required by OAuth. This OAuth request is used to invoke the API proxy operation.

---

### Scenario within API Connect

No changes to the definition of your API operation are required to support this scenario.

1. Alice grants permission for the application to access her information *before* the invocation of the API.
2. When the application provides the Authorization header, or query parameters, containing the OAuth details about the call to the operation endpoint, the header is automatically passed through to the target service without any additional configuration.

---

## OAuth introspection for third-party OAuth providers

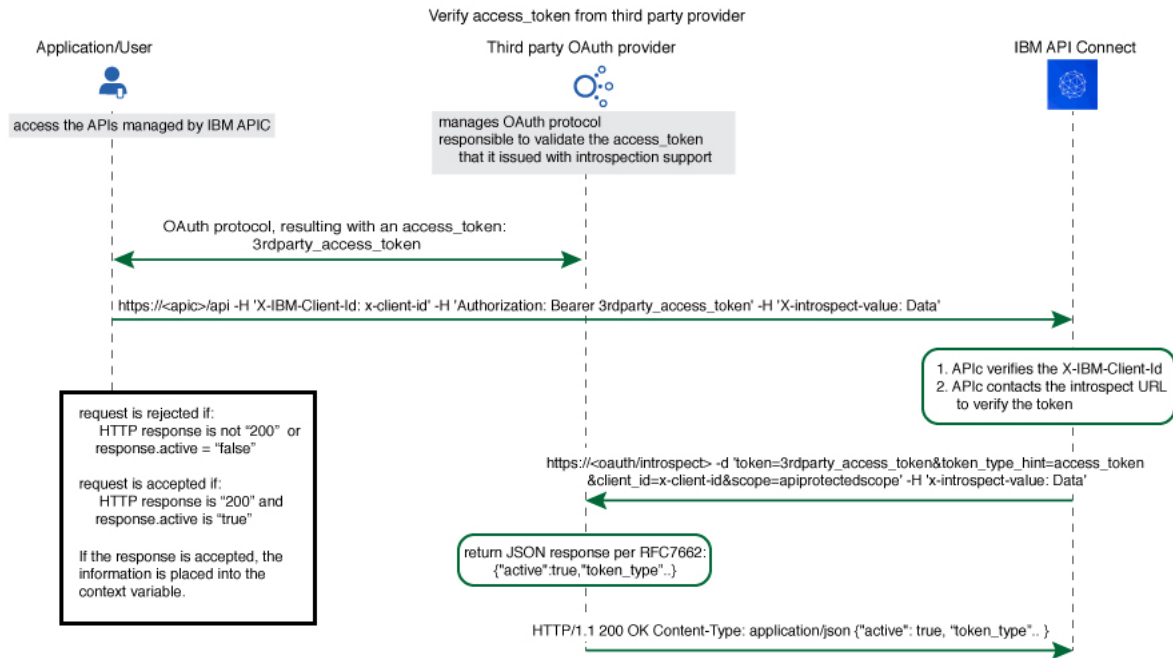
OAuth token validation can be offloaded to the third-party Open Authorization (OAuth)/Open ID Connect (OIDC) provider by using the Introspection URL. Clients can use a third-party OAuth or OIDC provider to obtain a token that is protected by IBM API Connect. IBM® API Connect can use this feature along with the mentioned provider to

protect access to the API.

You can use API Connect to protect an API that is secured by using the third-party OAuth access token in accordance with Introspection specification as defined in [RFC 7662](#). In addition to the specification, the `x-Introspect-` header is provided to pass other content to the third party as you require.

Authentication information can be carried in the request by configuring a basic authentication request header.

The following sequence diagram depicts the overall flow of request and response. The purpose of this diagram is only to provide a general visualization of the nature of the flow; for the precise details, refer to the explanatory information after the diagram.



The Introspect URL is configured in the Third Party OAuth provider configuration. See [Configuring a third-party OAuth provider](#).

When an API is protected by a third-party OAuth provider, API Connect will extract the bearer token and issue an HTTP POST request to the endpoint specified in the Introspect URL field.

The GET request is protected by API Connect with this feature.

You can use a header prefix to pass information to the third-party provider. The header prefix can include a regular expression and specifies the name pattern of the headers to use for sending additional information, such as `x-introspect-*`.

```

GET /petstore/pet/123 HTTP/1.1
Host: apiconnect.com
x-Introspect-type: dog
x-Introspect-name: simon
x-custom-apic: petstore123
Authorization: Bearer tGzv3JOkF0XG5Qx2TlKWIA
x-IBM-Client-Id: xxx-xxx

```

However, if you want to use a different header prefix, specify the required value in the Custom header pattern field in the third party OAuth provider configuration. For third-party OAuth provider configuration details, see [Configuring a third-party OAuth provider](#). The Custom header pattern feature is available only with the DataPower® API Gateway, if you are using the DataPower Gateway (v5 compatible) the header prefix must be `x-Introspect-`.

API Connect will issue this POST request to the introspection endpoint specified in `x-tokenIntrospect`, as illustrated in the code sample:

```

POST /oauth/introspectURL HTTP/1.1
Host: apiconnect.ibm.com
Content-Type: application/x-www-form-urlencoded
x-Introspect-type: dog
x-Introspect-name: simon
x-custom-apic: petstore123
token_type_hint=access_token&token=tGzv3JOkF0XG5Qx2TlKWIA

```

The third party OAuth/OIDC provider will respond with HTTP 200 indicating the request was successful and that payload contains the token information. API Connect honors the active claim as defined in the RFC specification.

If the value of the active claim is `true`, the token is treated as valid.

```

HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
 "active": true,
 "token_type": "bearer",
 "client_id": "xxx-xxx",
 "username": "John Smith",
 ...
}

```

If the value of the active claim is `false`, the token is treated as invalid.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
```

```
{
 "active":false
}
```

A response code other than **HTTP 200** indicates failure to process the request.

When the OAuth token is valid and active, context variables are populated with information from the introspect JSON response. For more information, see [API Connect context variables](#).

When contacting the introspection endpoint, API Connect uses `client_id/appId` and `client_secret/appSecret` to construct the HTTP Basic authorization header.

By default, if `x-introspect-basic-authorization-header` exists in the request, the value is used for the HTTP Basic authentication header when the introspection endpoint is contacted. API Connect verifies that the HTTP Basic authentication header value is Base64 encoded before it is sent, and encodes it if necessary as shown in the following example. If the header is already encoded, it is sent without modification.

```
GET /petstore/pet/123 HTTP/1.1
Host: apiconnect.com
x-introspect-basic-authorization-header: user:password
```

API Connect issues the following:

```
POST ..
Authorization: Basic M3JkLXBhcncR5LWNsaWVudF9pZDozcmQtcGFydH1fY2xpZW50X3NlY3JldA==token_type_hint=access_token&token= ..
```

If you are using the DataPower API Gateway, you can specify your own value for the HTTP Basic authentication header by providing the required value in the Basic authentication request header name field in the third party OAuth provider configuration. The following rules determine what authentication data is passed to the introspection endpoint:

- If there is a basic authentication header in the request, the specified credentials are used. The value must be either a string in the format `user:password`, or the Base64 encoded equivalent; API Connect will Base64 encode the value if necessary before sending the request to the introspection endpoint.
- If there is no basic authentication header in the request but there are values specified in the Basic authentication username and Basic authentication password fields in the third party OAuth provider configuration, those values are used.
- If there is no basic authentication header in the request, and user name and password details are not supplied in the third party OAuth provider configuration, but `client_id` and `client_secret` values are supplied in the body of the request, these are used.
- Otherwise, an error is returned.

For third-party OAuth provider configuration details, see [Configuring a third-party OAuth provider](#).

If either, or both, of `scope` and `scope validate url` are configured, and if the response is an active token with a scope claim from the third-party OAuth Provider introspection endpoint, API Connect will further enforce the scope validation in the following order:

1. If `scope` is configured for the OAuth API protection, verify the third-party scope against the scope that is configured.
2. If `scope validation url` is configured, verify the third-party scope against the scope validation URL.
3. If the third-party OAuth provider introspection endpoint does not return a scope, the Gateway will not check the scope defined in the security definition of the API where the third-party OAuth provider resource is used. For details on configuring an OAuth security definition in an API, see [Creating an OAuth security definition](#).

For more information, see [Scope](#).

DataPower Gateway (v5 compatible) only By default the API Connect client ID and scope are sent to the third party OAuth provider. You can suppress this behavior in either of the following ways:

- Supply a `suppress-parameters` header as follows:

```
suppress-parameters: client_id
```

```
suppress-parameters: scope
```

or

```
suppress-parameters: client_id scope
```

depending on which parameters you want to suppress.

- Define an API property called `suppress-parameters` in the API definition itself, with one of the following string values:

```
client_id
```

```
scope
```

or

```
client_id scope
```

depending on which parameters you want to suppress. For information on how to define API properties, see [Setting API properties](#).

---

## OAuth external URL and authentication URL

You can use the Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.

---

## Including custom metadata in a token

In many scenarios, custom metadata needs to be included during the access token generation process. The metadata is either stored inside the access token or it is sent along with the access token to the client application. The client application can then send that access token, or the metadata in the payload, in a subsequent request to IBM® API Connect where the metadata is retrieved, validated, or sent to the downstream systems as required.

Examples include, but are not limited to:

- When resource owners are authenticated, metadata about the authenticated resource owner needs to be stored within the access token.
- The grant type that was used to obtain the token is another example of metadata stored within the access token.
- A confirmation code that needs to be sent to the client application along with access token is stored as metadata within the access token.

## Configuring External URL or Authentication URL in API Connect to obtain metadata

Metadata can be set by using either or both of the following URLs:

- External URL - When you call the External URL, an HTTP GET request is sent and API Connect expects an HTTP 200 OK along with an optional set of the specified response headers.
- Authentication URL - When you call the Authentication URL, the API Connect gateway sends a GET request with HTTP headers and then processes any HTTP response from the URL. For authentication, a REST authentication service is expected at the Authentication URL.  
See: [Authentication URL](#).

The External URL endpoint is entered in the Metadata section when you configure an OAuth Provider in the Cloud Manager or API Manager UI.

Use the following HTTP headers in the response, depending on the type of gateway you are using:

- DataPower® API Gateway:

**X-API-OAUTH-METADATA-FOR-PAYLOAD**

**X-API-OAUTH-METADATA-FOR-ACCESSTOKEN**

Note: These are the default header names for the DataPower API Gateway but you can override them; see [Configuring metadata for a native OAuth provider](#).

- DataPower Gateway (v5 compatible):

**API-OAUTH-METADATA-FOR-PAYLOAD**

**API-OAUTH-METADATA-FOR-ACCESSTOKEN**

The response header value from **X-API-OAUTH-METADATA-FOR-PAYLOAD** or **API-OAUTH-METADATA-FOR-PAYLOAD** is placed in the response payload and indicated as **metadata**.

The response header value from **X-API-OAUTH-METADATA-FOR-ACCESSTOKEN** or **API-OAUTH-METADATA-FOR-ACCESSTOKEN** is placed within the access token and indicated as **miscinfo**.

The two metadata response headers are case insensitive and you must escape any special characters in the string value content.

An example response payload that contains metadata along with the access token:

```
{
 "token_type": "bearer",
 "access_token": "AAEkNzhjDHYyyYy...cL0Mv6ct137w7ZU",
 "metadata": "m:metadata-for-payload_content",
 "expires_in": 3600,
 "scope": "read",
 "refresh_token": "AAEnj5SynCMYbF...oEZ6JjxYax_HdNg",
}
```

This example output from the token introspection endpoint shows the contents of the access token with **"miscinfo"** containing the metadata information.

```
{
 "active": true,
 "token_type": "bearer",
 "client_id": "78c2f10f-799a-4e1f-8e0a-098634997a35",
 "username": "Fred Smith",
 "sub": "fred",
 "exp": 1479850049,
 "expstr": "2016-11-22T21:27:29Z",
 "iat": 1479846449,
 "nbf": 1479846449,
 "nbfstr": "2016-11-22T20:27:29Z",
 "scope": "read",
 "miscinfo": "m:metadata-for-access-token_content",
 "client_name": "MobileApp"
}
```

## Input to the External URL

The following request headers are sent to the External URL.

Note: Any existing metadata values that were previously sent from the Authentication URL are also sent in the two request headers **x-existing-metadata-for-payload** and **x-existing-metadata-for-access-token**. The Metadata URL can make use of this information to create a new set of metadata values.

The two request headers that are sent to the Metadata URL are displayed in bold text.

```
X-existing-metadata-for-payload payload-from-auth-url
X-existing-metadata-for-access-token token-from-auth-url
X-URI-in /org/env/miscinfo/oauth2/token (the URL that was sent to APICoconnect for this particular token request)
X-METHOD-in POST
X-POST-Body-in client_id=client_id&grant_type=password&scope=read&username=name&password=password
X-X-Client-IP IP_address
```

```
X-X-Global-Transaction-ID ID_number
...
```

Note: If you are using the DataPower API Gateway rather than the DataPower Gateway (v5 compatible), the `X-POST-Body-in` header is not supported.

## Retrieving Metadata in API Connect

As described in the previous example scenarios, the metadata can be retrieved from the access token in the Application API and sent to the downstream systems. Retrieval can be done in the API assembly, which is secured to accept tokens in the security definitions.

In the resource API that accepts an access token, the `miscinfo` field can be accessed in the Assembly with the `oauth.miscinfo` context variable, as in the example.

```
apim.setvariable('message.body', apim.getvariable('oauth.miscinfo'));
```

You can also use token introspection to look at the contents of the access token.

## Refresh tokens and metadata

The Authentication URL (if configured for authentication) is invoked first, during authentication of the resource owner. The External URL is invoked as the last step, just before the generation of the access token. The only exception is when access tokens are generated from a refresh token. In cases where refresh tokens are used to generate new access tokens, the External URL is not invoked. The metadata from the refresh token is retained and then copied into the newly generated access token.

## Identifying the source of the metadata

The metadata is prefixed with keywords to indicate whether it originated from the External URL or the Authentication URL.

- Metadata from the External URL is prefixed with `m`:
- Metadata from the Authentication URL is prefixed with `a`:

Note: When revocation is enabled, some internal details are also stored in the `miscinfo` field, in brackets within the access token as shown in the following example:

```
"miscinfo": "[tlsprofile@https://api-revoke-url:443/server/revocation-url]m:metadata-for-accesstoken_content"
```

## Maximum size of the metadata

Metadata for the access token cannot exceed 512 bytes.

Metadata for the payload does not have a specific size restriction, except for when you use the Authorization code grant type. These restrictions are described in following sections.

## Characters are not allowed in metadata in certain scenarios

When you use the Authorization code grant type, or when a consent form is used for implicit grant type, there is a temporary state or code where the metadata from the authentication URL is stored. API Connect internally uses two prefixes - `!ma` and `!mp` to differentiate between payload and token metadata received from the Authentication URL and store them internally in the temporary state/code. Hence these specific character sequences - `!ma` and `!mp` should not be used as the metadata itself.

## Grant types and metadata

The OAuth grant types described in the following sections include `authorization code` (`access code`), `implicit`, and `client credentials` (`application`).

Authorization code grant type

- When metadata is included from an Authentication URL for an Authorization code grant type, as it is a three legged flow, both the content and the payload are stored within the `dp-state` and carried on to the authorization code and to the access token. Note that around 10 characters are used internally to differentiate between the metadata for payload and metadata for the access token when stored in the `dp-state`. In addition, if revocation is enabled, that will also be part of the token metadata. Hence the combined size of the token metadata, the payload metadata (including the 10 characters of internal data), and internal revocation details, cannot exceed 512 bytes in total.

If the overall size of the metadata exceeds 512 bytes, then the access token generation succeeds, but the metadata fields contain an error message of "metadata too large" as shown in the example.

```
"metadata": "m:error: metadata too large for AZ code grant type[Authorization Code-metadata-url-payload]"
"miscinfo": "[r:gateway]m:error: metadata too large for AZ code grant type[Authorization Code-metadata-url-token]"
```

This size restriction can be overcome when the metadata is sent from the Metadata URL and not from the Authentication URL, because the metadata is not stored in `dp-state` or in the authorization code.

Example of the `authorization code` (`access code`) grant type:

```
$ curl -k -d
"grant_type=authorization_code&code=$mycode&client_id=$myid&scope=scope_introspect&redirect_uri="
https://9.70.153.90/fei/sb/introspectpl/oauth2/token
{ "token_type": "bearer",
 "access_token": "AAEk0Th1ZDhhNjYtYTQ1ZS00YTMzLWEON2QtZmE2OGZkMzQ0NzQ20ZN5T1_TqYFeIfB7BFf6HFgibEoOjWXXEA84oFsWuE4NY-
RRZVdnGSaXAIYJz7s5vczfk5EV-BIb_6P_1YKm3ahrfrhR5kI3sPO0uADEoseIP5-O9anUpEM5yhSayXvZbJ_6VDYz-hyXSJHTNqVj-
PHBialoRkBD5qca6k00fV2M", "metadata": "a:[Authorization Code-Test-auth-url-payload]", "expires_in": 3600,
 "scope": "scope_introspect", "refresh_token": "AAFAG1EVMbwicr_L0FTZ4q6HZ-
RcQygniXFC9zbSKO4wd3hcn1C4KQX21X0fL2c8cnmzCZgws8xxLzNyfjQhUJNG15C1GbTe3dwhXJdiWA0Go-
dduhVtCbG26sJRRXyYrMeRxxWnJsy1BETPI8HQEN4a_D7fmxKcTVRZBvq86byg95qe1ZKyERi0Lhxdd_04Nvss" }
```

```
$ curl -k -H "X-IBM-Client-Id: $myid" -H "X-IBM-Client-Secret: $mysecret" -d
"token_type_hint=access_token&token=$mytoken" https://9.70.153.90/fei/sb/introspectp1/oauth2/introspect
{ "active":true, "token_type":"bearer", "client_id":"98ed8a66-a45e-4a33-a47d-fa68fd344746", "username":"anyuser",
"sub":"anyuser", "exp":1484766368, "expstr":"2017-01-18T19:06:08Z", "iat":1484762768, "nbf":1484762768,
"nbfstr":"2017-01-18T18:06:08Z", "scope":"scope_introspect", "miscinfo":{"r:gateway}a:[Authorization Code-Test-auth-
url-token]", "client_name":"oauth_app" }
```

Implicit grant type

- When implicit grant type is used, the access token and the metadata are returned in the `Location` header as a fragment, as you see in the example.

```
< Location:
https://localhost#access_token=AAEkOThlZDhhNjYtYTQ1ZS00YTMzLWE0N2QtZmE2OGZkMzQ0NzQ2buS2KfWdq-
nYBJSi4nmPxQBtLael7tKBPRMzwP5BC386nlxpoOTE1G748ZVH6Mq_TJL3GeV3PtXTVIkWOLBJi_7t1jiQfnpVfrNkvvvZkhUexYmFkcDsmLSdaWxZ6Pc
IMPAC4ojT8qV1sYV-
ChTk36yQx_NiKimZaDikDk7WTg&expires_in=3600&scope=scope_introspect&token_type=bearer&metadata=a%3A[Implicit-Test-
auth-url-payload]

$ curl -k -H "X-IBM-Client-Id: $myid" -H "X-IBM-Client-Secret: $mysecret" -d
"token_type_hint=access_token&token=$mytoken" https://9.70.153.90/fei/sb/introspectp1/oauth2/introspect
{ "active":true, "token_type":"bearer", "client_id":"98ed8a66-a45e-4a33-a47d-fa68fd344746",
"username":"anyuser", "sub":"anyuser", "exp":1484768365, "expstr":"2017-01-18T19:39:25Z", "iat":1484764765,
"nbf":1484764765, "nbfstr":"2017-01-18T18:39:25Z", "scope":"scope_introspect", "miscinfo":{"r:gateway}a:[Implicit-
Test-auth-url-token]", "client_name":"oauth_app" }
```

Client credentials grant type

Authentication URL will not be invoked when using client credentials grant type, as there is no resource owner. The metadata from Authentication URL is not available for this grant type. However, content returned from Metadata URL will be included as metadata.

## Behavior when retrieving metadata with both an External URL and an Authentication URL

If an External URL is configured and a connection to the external server is successful, the response headers overwrite any existing metadata obtained from the Authentication URL to become the final value. Therefore, you must carefully examine the incoming request headers and create appropriate response headers from the External URL.

If an External URL is configured, but the connection to the External URL fails, then a failure message of "error on metadata url" is written for metadata in both the payload and the access token.

If an External URL is configured and the connection is successful, but the remote server does not send any of the specified HTTP response headers, a blank value is written for metadata in both the payload and the access token.

Attention: An External URL overwrites existing values from Authentication URL. This includes blank values.

If no External URL is configured, the metadata that is obtained from the Authorization URL is retained as the final value.

## Scope

Per IETF RFC 6749, the value of the scope parameter is expressed as a list of space-delimited, case-sensitive strings.

In IBM® API Connect, scopes have no inherent meaning. Instead, scopes are defined in the OAuth Provider so that an application can request an access token that is valid for one or more of the scopes. In the secured API, scopes are listed as requirements for an access token to be considered valid. All scopes that are listed by the security definition for the API must be granted by the access token.

Note:

- An OAuth Provider must have at least one scope.
- An OAuth token will not be granted unless a default scope is configured at the provider, or one or more predefined scopes are included in the grant request.
- If no default scope is defined, and the request does not contain a scope, an invalid scope error is returned for the request.

## OAuth provider

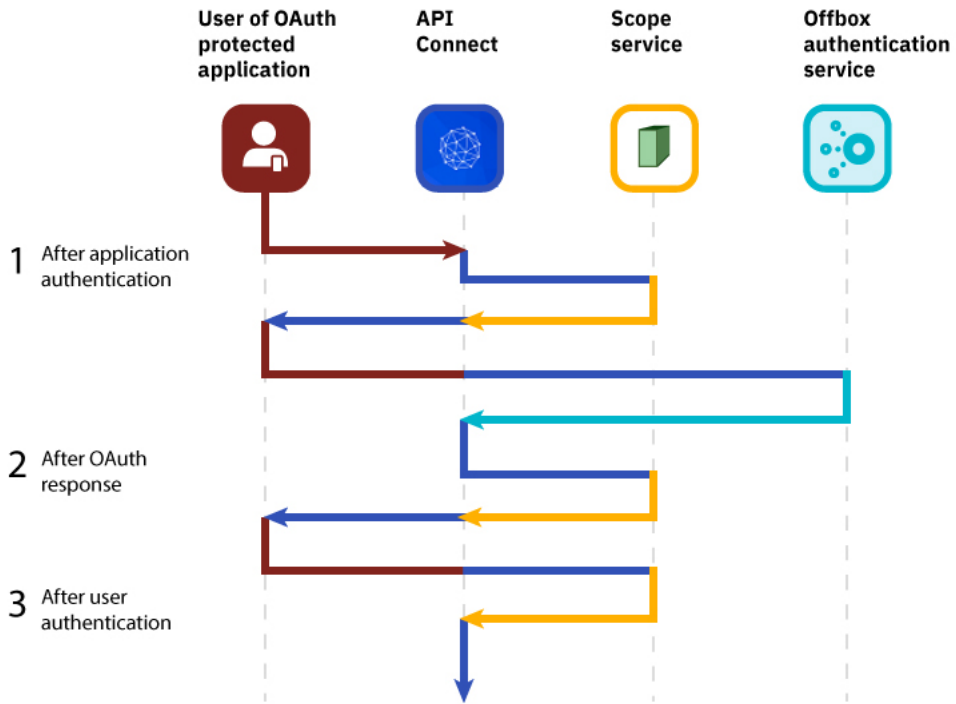
To provide more refined support for the OAuth scope handling, API Connect allows the [Authentication URL user registry](#) extension to modify the scope value.

When you define an [OAuth provider](#), the Advanced scope check extensions provide the flexibility to check and override allowed scopes. The optional extensions are Application scope check and Owner scope check.

The scope that is eventually received by the application is determined by the interactions that are described in the three following paragraphs. Scope processing follows the sequence of items 1, 2, and 3 in the following list, offering three opportunities to override the scope value. Figure 1 provides an overview of the process.

- After the application successfully authenticates, and if OAuth Native provider `>Advanced scope check` `>Application scope check` is configured, API Connect makes a call to allow extra verification and uses the contents of `x-selected-scope` to override the scope value that was initially requested by the application. When Application scope check is enabled, the HTTP response header `x-selected-scope` must be present, or the call fails.
- If OAuth Native provider `>User Security` `>Authentication` is configured to authenticate application users using an Authentication URL, then API Connect makes a call, as documented in [Authentication URL user registry](#). When the response code is HTTP 200, and the response header `x-selected-scope` is present, the value that is configured in `x-selected-scope` is used as the new scope value, overriding both what the application already requested and what was provided in the Application scope check described in paragraph 1. In the response header, `x-selected-scope` is an optional element.
- After the user successfully authenticates, and if OAuth Native provider `>Advanced scope check` `>Owner scope check` is enabled and configured with a valid URL, API Connect makes a call to allow the content of `x-selected-scope` to refine the scope value. When Owner scope check is enabled, the HTTP response header `x-selected-scope` must be present, or the call fails.

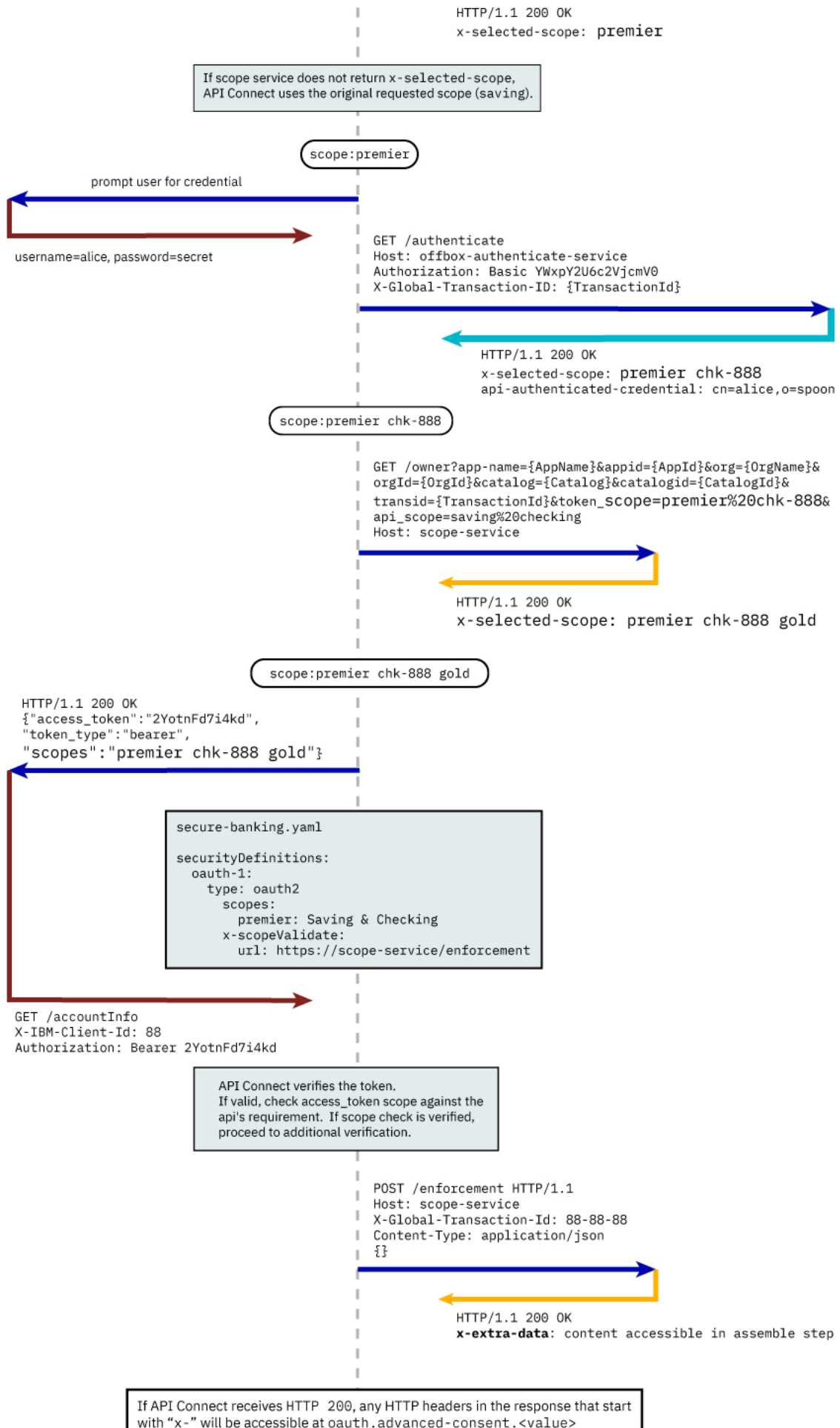
Figure 1. OAuth advanced scope overview



The final scope permission that is granted by the access token is the result of the flow described in items 1, 2, and 3. [Figure 2](#) shows a more detailed view of the transaction flow with examples that show when `x-selected-scope` provides a new scope value.

Figure 2. OAuth advanced scope detail







For the above example, `x-extra-data` is accessible at `oauth.advanced-consent.x-extra-data`.  
If other than `HTTP 200` is returned, it has the same effect as if the access token does not contain necessary permissions to access the resource.

## Consumer API enforcement

Standard scope validation

To access the API `/getaccount` the application must send a `GET` request with an access token that contains the scope, or scopes, defined in the OAuth provider. If the request doesn't contain a scope, you must specify a default scope to use. If no default scope is defined, and the request doesn't contain a scope, an invalid scope error is returned for the request.

```
GET /getaccount
HTTP/1.1
Host: server.example.com
X-IBM-Client-Id: 8888-8888-8888
Authorization: Bearer AAEkNjVkwOWIyYjgtOWY5ZS00YWQwLWIyYzktZ
```

The following application OpenAPI file `secure-banking.yaml` defines the scope, or scopes, that must exist in the token to be granted access to the API `/getaccount`.

```
secure-banking.yaml

securityDefinitions:
 scope-only:
 type: oauth2
 description: ''
 flow: implicit
 authorizationUrl: ''
 scopes:
 checking: 'Checking Account'
 saving: 'Saving Account'
 mutual: 'Mutual Fund Account'
 security:
 - scope-only:
 - checking
 - scope-only:
 - saving
 - mutual
```

In the examples, the access token `AAEkNjVkwOWIyYjgtOWY5ZS00YWQwLWIyYzktZ` is able to access the API because it contains one, or a combination of `scope-only` that is defined in `secure-banking.yaml` such as:

- `checking`
- `saving mutual`
- `checking saving mutual`

Advanced Scope Check

Administrators can enable an additional scope check by configuring the consumer API property Advanced Scope Check URL that becomes `x-scopeValidate` as shown in the following OpenAPI file example.

```
securityDefinitions:
 advanced-scope-only:
 type: oauth2
 description: ''
 flow: implicit
 authorizationUrl: ''
 scopes:
 checking: 'Checking Account'
 saving: 'Saving Account'
 mutual: 'Mutual Fund Account'
 x-scopeValidate:
 url: 'https://advanced-scope-check.bk.com/validate-scope'
 tls-profile: 'ssl-client'
```

After API Connect successfully verifies the access token against any scope requirement, API Connect will make an `HTTP POST` request to the `x-scopeValidate` endpoint similar to the following example. Response code `HTTP 200` from the endpoint indicates a success. Any other response code, or a connection error, is treated as a permission error for the token.

```
POST /validate-scope?app-name=..&appid=..&org=..&orgid=..&catalog=..&catalogid=..&transid=..
HTTP/1.1
Host: advanced-scope-check.bk.com
Content-Type: application/json
```

```
{
 "context-root" : checking,
 "resource" : accountinfo,
 "method" : GET,
 "api-scope-required" : [jointaccount],
 "access_token" : {
 "client_id" : "2cd71759-1003-4a1e-becb-0474d73455f3",
 "not_after" : 174364070,
 "not_after_text" : "2017-07-11T02:27:50Z",
 "not_before" : 174360470,
 "not_before_text" : "2017-07-11T01:27:50Z",
 "grant_type" : "code",
 "consented_on" : 1499736470,
 "consented_on_text" : "2059-07-11T01:27:50Z",
 "resource_owner" : "cn=spoon,email=spoon@poon.com",
 "scope" : "jointaccount mutual",
 "miscinfo" : "[r:gateway]"
 }
}
```

An example of successful response follows.

```
HTTP/1.1 200 OK
Cache-Control: no-store
Pragma: no-cache
X-Custom-For-Assemble-Process: audit
```

Any HTTP response header that begins with "x-" is kept as the context variable `oauth.advanced-consent`. Based on the example successful response, `X-Custom-For-Assemble-Process: audit` becomes `oauth.advanced-consent.x-custom-for-assemble-process`, and can be accessed in the assemble step.

Additional validation options

You can optionally use additional fields for validation:

Request Header

Defines the regular expression to match against **request** headers. Matching headers are included in the request to the advanced scope validation endpoint.

Response Context Variable

Defines the regular expression to match against **response** headers. Matching headers are saved as context variables in the format `oauth.advanced-consent.*`.

## Related information

---

- [IETF RFC 6749](#)

## Tokens

Tokens can be managed using refresh tokens and revocation URLs. You can extend the life of tokens using refresh tokens. You can end the life of a token by specifying a revocation URL.

This section contains information regarding token management, including refreshing and revoking tokens, redirecting, and managing tokens with the DataPower Gateway.

- [Refresh tokens](#)  
If you are using OAuth authentication, you can enable refresh tokens. Refresh tokens are issued to the client to obtain a new access token when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or narrower scope.
- [Token revocation](#)  
In IBM® API Connect, you can revoke or refresh tokens. If necessary, you can also revoke all tokens that are issued to a specific client ID or a resource owner.
- [Authenticating and authorizing through a redirect URL](#)  
You can use a service that is hosted externally from IBM API Connect to collect authentication and authorization details from your user when an application requests access on that user's behalf.
- [Token management with the DataPower Gateway \(v5 compatible\)](#)  
API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.
- [Token management with the DataPower API Gateway](#)  
API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.

## Refresh tokens

If you are using OAuth authentication, you can enable refresh tokens. Refresh tokens are issued to the client to obtain a new access token when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or narrower scope.

When you are using OAuth authentication, API requests must include a valid access token, by using the Authorization HTTP header. Access tokens that are issued by the IBM® API Connect Token Endpoint are valid for 3600 seconds (1 hour) by default, as indicated by the `expires_in` property that is returned on the token request. The following code block shows an example API request with an Authorization header:

```
GET /bankingApi/accountSummary?client_id=32427ce5-bb7c-48a7-9de3-4bb629091103
HTTP/1.1
Accept: application/json
Host: api.ibm.com
Authorization: Bearer AAEFYy1hbGx1hdS5nVX4x6iTL2sb3ymBivQb...
```

After an access token expires, if the option is enabled in the OAuth provider configuration Tokens > Refresh tokens screen, the application uses refresh tokens. Each refresh token is valid for approximately 31 days after it is issued (or for the Time to Live time period specified) and can be used only once to request a new access token. Along with the new access token, a new refresh token is also returned. For details on how to enable refresh tokens, see [Configuring a native OAuth provider](#).

If the access token is expired and the application does not have a refresh token, it must restart the OAuth exchange by using the choice of Grant Type(s) allowed by the OAuth provider.

## Token revocation

In IBM® API Connect, you can revoke or refresh tokens. If necessary, you can also revoke all tokens that are issued to a specific client ID or a resource owner.

## Token revocation using an external third party service

This topic describes token revocation using a third party, external service. This option is configured in the Native OAuth provider, using the Token Management > Type = Third party screen. The revocation URL is an endpoint that links to an external service which contains information about access or refresh tokens. API Connect is involved in the initial creation and validation of tokens. When an OAuth revocation URL is present, API Connect calls the URL to determine if the associated token can be trusted.

The token server then checks a token *blocklist* (a data store of inactive tokens) to ensure that the token is still valid. If the token is still valid, API Connect continues the processing.

## Examples

---

A number of revocation examples follow. The first shows a sample fetch request and the response from a remote revocation URL.

### GET request and response

---

In this example, an API Gateway server issues a GET request to the Revocation URL and receives a result. It shows that different resource owners (Laura and Emily) can revoke all tokens when using the same application (client ID).

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
GET <revocationURL> HTTP/1.1
User-Agent: IBM-APIManagement/4.0
Accept: application/xml; text/xml
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-revocation>

<!--
Access Tokens and/or Refresh Tokens that are revoked can be individually listed.
To keep this list small, please only include access tokens and refresh tokens that are valid.
For access tokens, any token older than 20 minutes is no longer valid.
For refresh tokens, any token older than 44700 minutes is no longer valid.
-->

<token type="access">AAETb2F1dGgtcmV2b2t1LWN1c3RvbfZaRlVbnPSc1...</token>
<token type="refresh">fZaRlVbnPSc1UGTjCRdq4mPbOosD2+aZIKbJ6bTeW...</token>

<!--
If a resource owner has revoked all tokens issued to a given application, please
list them as shown here.
-->

<resource-owner client-id="760d75a2-44b1-4485-8c6f-0d264fcf7398">laura</resource-owner>
<resource-owner client-id="760d75a2-44b1-4485-8c6f-0d264fcf7398">emily</resource-owner>

</oauth-revocation>
```

### POST request and response

---

The next example shows a post request and response.

Request:

```
POST <revocationURL> HTTP/1.1
User-Agent: IBM-APIManagement/4.0
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<token>
 <token_type>bearer</token_type>
 <access_token>AAENYy1hbGwtcmVmcmVzaOfNeQKX8ZeojsBY9v0FI7/OerQvzKHq...</access_token>
 <expires_in>3600</expires_in>
 <scope>3600</scope>
 <resource-owner>alice</resource-owner>
 <client_id>83d9cdcd-ba72-4d00-abae-005da8da5fb1</client_id>
</token>
```

Response:

```
HTTP/1.1 200 OK
```

### Provide token information on revocation request

---

In this example, the application calls an API and passes a bearer token. In response, the Gateway fetches the revocation URL and provides information on the token being verified.

Gateway:

```
GET <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2t1LWN1c3RvbfZaRlVbnPSc1
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Alternatively, the same process occurs when using a refresh token to issue a new access token. The application sends a refresh request to the token service. The Gateway then fetches the revocation URL, providing information on the refresh token being verified.

Gateway:

```
GET <revocationURL>GET HTTP/1.1
refresh-token: AAETb2F1dGgtcmV2b2t1LWN1c3RvbfZaRlVbnPSc1
```

```
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Note: If you are using a third party OAuth provider then, for API Calls with bearer tokens, when Introspect URL is enabled on the API, the revocation URL is not applicable. Instead, the third party endpoint must validate the token and also check for revocation before returning a 200 OK response to the Gateway.

## Revoking tokens issued to Alice up to and including a specific date

On May 1st, Alice loses her phone and needs to reset her password. As a result, the token provider wants to revoke every token issued before Alice lost her phone. In this example, the Gateway sends a GET request to the revocation service. The revocation service replies confirming revocation of the specified tokens.

Gateway:

```
GET <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2t1LWN1c3RvbfZaRlVbnPSc1
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Revocation Service:

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-revocation>
 <resource-owner before="2015-05-01T09:30:10Z">alice</resource-owner>
</oauth-revocation>
```

## Revoking all tokens issued up to and including a specific date

Under certain catastrophic conditions, you may need to revoke all tokens issued up to and including a specific date, for example, May 1st. In this example, the Gateway sends a GET request to the revocation service. The revocation service replies confirming revocation of the specified tokens.

Gateway:

```
GET <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2t1LWN1c3RvbfZaRlVbnPSc1
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Revocation Service:

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-revocation>
 <everytoken before="2015-05-01T09:30:10Z" />
</oauth-revocation>
```

## Putting it all together

The following shows the examples contained in this topic executed in a single action:

Gateway:

```
GET <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2t1LWN1c3RvbfZaRlVbnPSc1
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Revocation Service:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Cache-Control: public, max-age=120
Date: Fri, 08 May 2015 21:49:03 GMT
```

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-revocation>
 <resource-owner before="2015-04-08T09:30:10Z">mary</resource-owner>
 <resource-owner before="2015-04-12T09:30:10Z">john</resource-owner>
 <resource-owner before="2015-04-13T09:30:10Z">kevin</resource-owner>
 <resource-owner before="2015-04-01T09:30:10Z">alice</resource-owner>
</oauth-revocation>
```

Notes:

- In the previous example, there are no entries older than one month in the response (the maximum life of a refresh token).
- Each response is cached for up to two minutes according to response's directive.
- The **before** attribute uses the **xs:dateTime** syntax.

## Authenticating and authorizing through a redirect URL

You can use a service that is hosted externally from IBM® API Connect to collect authentication and authorization details from your user when an application requests access on that user's behalf.

## Before you begin

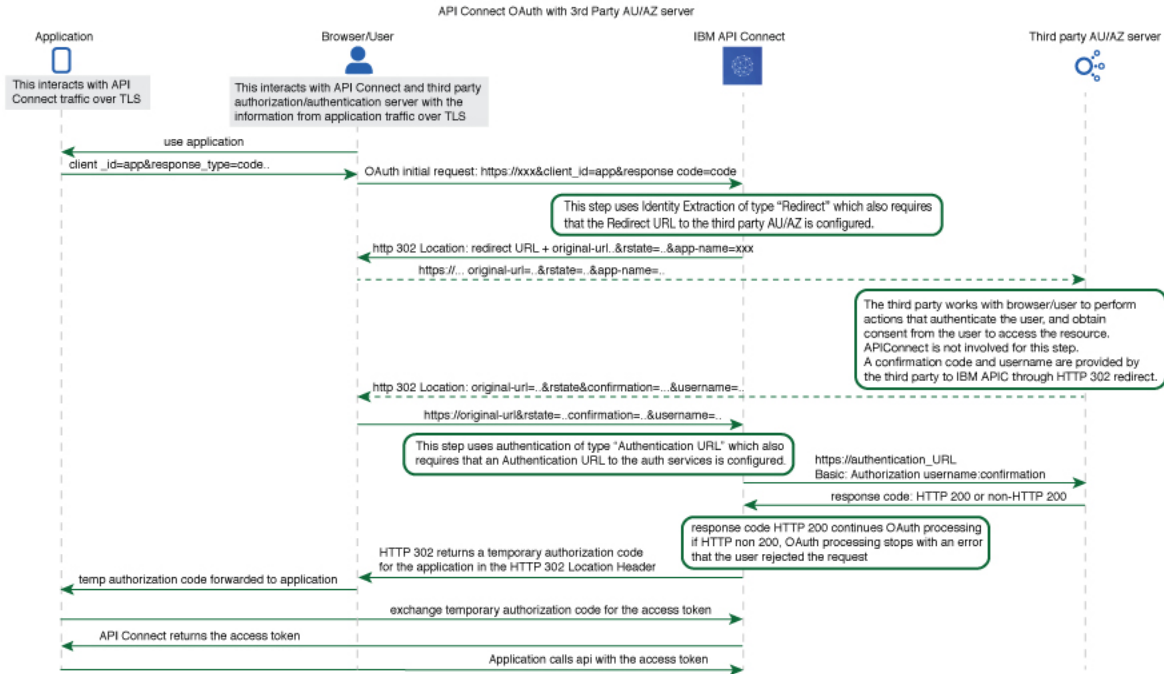
To complete this task, you will need to either create or have created an OAuth security definition that uses Implicit grant type or Access Code (Authorization Code) grant type. For more information, see [Creating an OAuth security definition](#).

## About this task

If you use methods for authentication that are not supported by API Connect, you can redirect users to a suitable URL at which they can authenticate. The user is then returned to the OAuth process after authentication and authorization have been confirmed.

The following illustration indicates the transaction flow for third party authentication.

Figure 1. Third party authentication (AU) and authorization (AZ) transaction flow



1. The application initiates a request to access an API protected with a third-party entity. API Connect redirects the application with an **HTTP 302** redirect based on **identity extraction** -> **redirect** -> **redirect-url**, for user authentication (and optional authorization).
  2. The application communicates directly with the third-party entity to gather user identity. API Connect is not involved in this communication. After the third-party entity finishes processing authentication (and optional authorization), it returns an **HTTP 302** redirect that uses the **original-url** from the request, with the username and confirmation code appended.
  3. API Connect receives the request that includes the username and confirmation code, and communicates with the authentication URL, based on **authentication** -> **x-ibm-authentication-url**, to confirm user identity before the request is completed.
  4. An **HTTP 200** response from the third-party entity allows API Connect to continue the OAuth 2.0 request process as if the owner is authenticated. The request is then processed according to the **grants** type.
    - - **accessCode** returns a temporary code to the application.
    - - **implicit** returns the access token to the application.
- For any response other than HTTP 200, the request fails with a statement added to the error log.

## Procedure

To create an external form, and to indicate the URL to which API Connect will redirect users, complete the following instructions:

1. Create your service for authentication and authorization. You will use the URL of the landing page of this service as your redirect URL.
  - a. To include elements in your form that are provided by API Connect, use the following query parameters from the URL that your user is redirected to. Note: With the exception of **original-url**, none of these parameters are included in the URL automatically; you must add them as required.

**app-name**  
The name of the application requesting access, as provided through the Developer Portal.

**appid**  
The id of the application requesting access.

**catalog**  
The name of the catalog where the product is being used by the application.

**catalogid**  
The id of the catalog where the product is being used by the application.

**catalogtitle**  
User-friendly display name for the catalog.

**org**  
The name of the consumer organization that hosts the application.

**orgid**  
The id of the consumer organization that hosts the application.

**orgtitle**  
User-friendly display name for the organization.

**original-url**

The original URL that the user was directed to by the application, including query parameters from the original URL that are necessary for standard OAuth 2.0 requests. You can include these parameters in your service to provide information to the user. Additionally the `state_nonce` is appended. The `state_nonce` is a hash code generated by API Connect for verification purposes. The URL is URL-encoded and should be decoded before further use, the `state_nonce` should remain unchanged.

provider

The name of the API provider organization.

providerid

The id of the API provider organization.

providertitle

User-friendly display name for the provider organization.

requested-scope

[optional] If [Application Scope check](#) is enabled and replaces the `scope` from the initial application request, this field holds the `scope` value from the initial application request, and the new replacement scope value is put into `original-url`.

transid

transaction id used in the GW for the transaction which trigger this call

The URL to which the user is sent to when they are redirected to your page has the following form:

**`Redirect_URL?original-url=Original_URL&state_nonce=R_State&app-name=Application_Name`**

where all variables are as described previously. The Redirect URL does not have a size limit enforced by API Connect.

- b. Create the stages of authentication, authorization, and any intermediate stages that you require to take place before you allow access to the application. Upon completion of these stages, redirect the user to the `Original_URL` and append a user name, their confirmation code, and the application name to be evaluated for access grant or denial by API Connect. The confirmation code does not have a size limit enforced by API Connect.

Original URL requires the following form:

**`Original_URL&username=User_Name&confirmation=Confirmation_Code`**

where all variables are as described previously.

For example:

```
https://your_gateway.com/your_org/your_catalog/your_api/oauth/authorize?
response_type=code&redirect_uri=https://example.com/redirect&scope=/your_api&client_id=5af57a4a-6db9-4141-ad08-
5709432af66e&state_nonce=HoIbRG+6bZtq1B7LDkq4gj1D3SHKglCbnYdHs/bMz2Y=&username=spoon&confirmation=12345678
```

- c. To send your own error responses after the authentication and authorization service, redirect the user to the `Original_URL` and append an error code. You can also append a user name. Use the following form:

**`Original_URL&username=User_Name&error=Error_Response`**

where `Error_Response` is the message you wish to send and all other variables are described as previously.

For example:

```
https://your_gateway.com/your_org/your_catalog/your_api/oauth/authorize?
response_type=code&redirect_uri=https://example.com/redirect&scope=/your_api&client_id=5af57a4a-6db9-4141-ad08-
5709432af66e&state_nonce=HoIbRG+6bZtq1B7LDkq4gj1D3SHKglCbnYdHs/bMz2Y=&username=User&error=access_denied
```

2. Create a service to validate the confirmation code and user name. API Connect makes a GET call to your authentication URL after the user is redirected back to the authorization URL. When the call is made, it includes in its authorization header the user name and confirmation code you supplied previously. Confirm that these are correct and respond with an HTTP success code such as 200 OK if you want to allow access, or non-200 HTTP response code, such as 401 Unauthorized to deny access.
3. In your OAuth provider configuration, supply the redirect URL that is used in Step 1 and the authentication URL that is used in Step 2. For more information on configuring an OAuth Provider, see [Configuring a native OAuth provider](#) when using API Manager or [Configuring a native OAuth provider](#) when using Cloud Manager.

---

## Token management with the DataPower Gateway (v5 compatible)

API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.

In order to manage tokens with the DataPower® Gateway (v5 compatible), you must set the Token Management Type to Native in your Native OAuth provider configuration. See [Configuring token management and revocation for a native OAuth provider](#) when using Cloud Manager or [Configuring token management and revocation for a native OAuth provider](#) when using API Manager.

Also, the DataPower quota enforcement server must be enabled on the DataPower Gateway to use the distributed cache to manage tokens. See [Quota enforcement](#).

When distributed cache support is enabled, replay protection is provided across the gateway cluster through the quota enforcement server. This support ensures that the same token cannot be reused across the members of the quota enforcement peer group.

Important:

If you have a cluster of DataPower Gateway servers, the OAuth data synchronization behavior across the servers depends on whether or not you enable revocation:

- If you enable revocation, API Connect uses the DataPower quota enforcement server, and OAuth data is synchronized across the servers. If an access token is obtained from one server, the OAuth data synchronization ensures that the same authorization code cannot be used to obtain an access code from another server. You must ensure that the DataPower quota enforcement server is configured.
- If you disable revocation, API Connect does **not** use the DataPower quota enforcement server and OAuth data does not synchronize across the cluster of DataPower Gateway servers. Therefore, to prevent the same authorization code being used to obtain an access code from more than one server you must configure DataPower to synchronize OAuth data across the servers, by using the DataPower Gateway console.

To configure DataPower to synchronize OAuth data, ensure that the DataPower quota enforcement server is configured. For more information, see [Configuring the quota enforcement server](#).

## Resource owner revocation

When Resource owner revocation path is selected in the Token Management screen, the configuration inserts two REST API calls to /oauth2/issued.

- An HTTP GET operation that retrieves a list of all granted permissions for a specific user.
- An HTTP DELETE operation that revokes an application for a specific user.

The setting inserts header-based security definitions of client ID and client secret as shown in the **View permissions example**. The API call to revoke a given application for a given user is shown in the **Revoke permissions example**.

View permissions example

To list all the applications granted by user `cn=spoon,o=ibm` with username `spoon` and password `spoon` using a registered administration application of `5287fe53-8747-438a-8262-681ec75b79c5`.

- Request:

```
GET /oauth2/issued
HTTP/1.1
Host: apic.ibm.com
x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
Authorization: Basic c3Bvb246c3Bvb24=
```

- Response:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
[
 {
 "clientId": "7369ad66-5674-b7d3-4567-de35283421aca",
 "owner": "cn=spoon,o=ibm",
 "clientName": "PetStore Application",
 "scope": "listpet",
 "issuedAt": 1503327054,
 "consentedOn": 1503327054,
 "expiredAt": 1503330654,
 "refreshTokenIssued": false,
 "appId": "d2031f0f27339315333734ab9",
 "org": "PetStoreOrg",
 "orgTitle": "Katie Pet Grooming Inc",
 "orgId": "5887803de4b06e6998c4b2c7",
 "provider": "SuperStore",
 "providerTitle": "Simon SuperStore",
 "providerId": "5887803de4b06e6998c4b2c7",
 "catalog": "publicapi",
 "catalogTitle": "For public",
 "catalogId": "5887803de4b06e6998c4b2d3"
 },
 {
 "clientId": "a8746323-9825-a842-8736-abd8202356ac8",
 "owner": "cn=spoon,o=ibm",
 ...
 }
]
```

Revoke permissions example

To revoke application `a8746323-9825-a842-8736-abd8202356ac8` by owner `cn=spoon,o=ibm`.

- Request

```
DELETE /oauth2/issued?client-id=a8746323-9825-a842-8736-abd8202356ac8
HTTP/1.1
Host: apic.ibm.com
x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
Authorization: Basic c3Bvb246c3Bvb24=
```

- Response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache
{ "status": "success" }
```

## Client revocation

When Client revocation path is selected in the Token Management screen, the result is the following:

This option inserts one REST API call to /oauth2/revoke, which supports OAuth 2.0 [IETF RFC 7009](#). An HTTP POST operation that an application can send to this API to revoke either an `access_token`, or `refresh_token` with `token_type_hint` as shown in the following examples:

Revoke `access_token`

- Request:

```
POST /oauth2/revoke
HTTP/1.1
```

```
Host: apic.ibm.com

x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iNluU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
Content-Type: application/x-www-form-urlencoded

token_type_hint=access_token&token=AAIHZGvmYXvsdD1-KqwD0Yc3EDn941SWX14xuR...
```

• Response:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache

{ "status": "success" }
```

Revoke refresh\_token

• Request:

```
POST /oauth2/ revoke
HTTP/1.1

Host: apic.ibm.com

x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iNluU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
Content-Type: application/x-www-form-urlencoded

token_type_hint=refresh_token&token=.....
```

• Response:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache

{ "status": "success" }
```

---

## Token management with the DataPower API Gateway

API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.

Note: The steps described in this topic are required if you are using the DataPower® OVA or appliance version of the gateway. For the API Connect supplied Kubernetes deployment, the API Security Token Manager is preconfigured provided that, in the gateway service custom resource file, v5 compatibility mode is disabled and the token management service is enabled:

```
.
.
.
spec:
.
.
.
.
.
.
apicGatewayServiceV5CompatibilityMode: false
tokenManagementService:
 enabled: true
.
.
.
```

For more information, see [Installing the DataPower Gateway subsystem](#).

In order to manage tokens with the DataPower API Gateway, you must set the Token Management Type to Native in your Native OAuth provider configuration. See [Configuring token management and revocation for a native OAuth provider](#) when using Cloud Manager or [Configuring token management and revocation for a native OAuth provider](#) when using API Manager.

Note: This setting is also required when you use a third-party OAuth provider and **Introspect cache type** is set to **protocol** or **Time to live**.

For token management with the DataPower API Gateway, you must configure the API Security Token Manager on the gateway. Complete the following steps:

1. Log in to the DataPower administration console, selecting apiconnect for the domain, and WebGUI for the Graphical Interface.
2. In the search box, enter API Security Token Manager, then click on the API Security Token Manager link that displays in the search results.
3. For the Administrative state, select enabled.
4. Click the + icon alongside the Gateway Peering field to create a new gateway peering object.
5. Provide a Name and a Local address.
6. In the Local port and Monitor port fields, provide values that aren't already in use by services for the Local address.
7. Ensure that Peer group mode is selected.
8. Alongside the Peers field, use the add button and accompanying field to add at least two peers, to ensure that quorum is achieved.
9. If the Enable SSL option is selected, select a value for the Identification credentials.
10. For Persistence location, select a setting other than memory.
11. When done, click Apply.
12. Repeat steps 1 to 11 for each DataPower API Gateway in the peer group.
13. When done, click Apply, then click Save Configuration to save your changes.



## Resource owner revocation

When Resource owner revocation path is selected in the Token Management screen, the configuration inserts two REST API calls to /oauth2/issued.

- An HTTP GET operation that retrieves a list of all granted permissions for a specific user.
- An HTTP DELETE operation that revokes an application for a specific user.

The setting inserts header-based security definitions of client ID and client secret as shown in the **View permissions example**. The API call to revoke a given application for a given user is shown in the **Revoke permissions example**.

View permissions example

To list all the applications granted by user `cn=spoon,o=ibm` with username `spoon` and password `spoon` using a registered administration application of `5287fe53-8747-438a-8262-681ec75b79c5`.

- Request:

```
GET /oauth2/issued
HTTP/1.1
Host: apic.ibm.com
x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
Authorization: Basic c3Bvb246c3Bvb24=
```

- Response:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
[
 {
 "clientId": "7369ad66-5674-b7d3-4567-de35283421aca",
 "owner": "cn=spoon,o=ibm",
 "clientName": "PetStore Application",
 "scope": "listpet",
 "issuedAt": 1503327054,
 "consentedOn": 1503327054,
 "expiredAt": 1503330654,
 "refreshTokenIssued": false,
 "appId": "d2031f0f27339315333734ab9",
 "org": "PetStoreOrg",
 "orgTitle": "Katie Pet Grooming Inc",
 "orgId": "5887803de4b06e6998c4b2c7",
 "provider": "SuperStore",
 "providerTitle": "Simon SuperStore",
 "providerId": "5887803de4b06e6998c4b2c7",
 "catalog": "publicapi",
 "catalogTitle": "For public",
 "catalogId": "5887803de4b06e6998c4b2d3"
 },
 {
 "clientId": "a8746323-9825-a842-8736-abd8202356ac8",
 "owner": "cn=spoon,o=ibm",
 ...
 }
]
```

Revoke permissions example

To revoke application `a8746323-9825-a842-8736-abd8202356ac8` by owner `cn=spoon,o=ibm`.

- Request

```
DELETE /oauth2/issued?client-id=a8746323-9825-a842-8736-abd8202356ac8
HTTP/1.1
Host: apic.ibm.com
x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
Authorization: Basic c3Bvb246c3Bvb24=
```

- Response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache
{ "status": "success" }
```

## Client revocation

When Client revocation path is selected in the Token Management screen, the result is the following:

This option inserts one REST API call to /oauth2/revoke, which supports OAuth 2.0 [IETF RFC 7009](#). An HTTP POST operation that an application can send to this API to revoke either an `access_token`, or `refresh_token` with `token_type_hint` as shown in the following examples:

Revoke `access_token`

- Request:

```

POST /oauth2/revoke
HTTP/1.1

Host: apic.ibm.com

x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iNluU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
Content-Type: application/x-www-form-urlencoded

token_type_hint=access_token&token=AAIHZGVmYXVsdD1-KqwD0Yc3EDn94lSWX14xuR....

```

- Response:

```

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache

{ "status": "success" }

```

Revoke `refresh_token`

- Request:

```

POST /oauth2/revoke
HTTP/1.1

Host: apic.ibm.com

x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iNluU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
Content-Type: application/x-www-form-urlencoded

token_type_hint=refresh_token&token=.....

```

- Response:

```

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache

{ "status": "success" }

```

## Authentication URL user registry

You can use an Authentication URL user registry to specify a REST authentication service that manages user authentication, and optionally provides additional metadata to be embedded in the token.

This support can optionally enable any of the following:

- Providing the authenticated credential to IBM® API Connect. For example, the user logs-in with user name: `spoon`, and password: `fork`. When the user is authenticated, the credential becomes `cn=spoon,o=eatery`. The credential is kept in the OAuth `access_token` to represent the user.
- Providing metadata support. Allow extra metadata to be stored in the `access_token`.
- Overriding the `scope` that the application receives after a successful OAuth protocol processing. By responding with a specific header, the Authentication URL endpoint can replace the `scope` value that the application receives. For example, you can provide a specific resource owner an account number within the `scope` header response for use in future processing steps.

When you call the Authentication URL user registry, the API Connect gateway sends a GET request with HTTP headers and then processes any HTTP response from the URL. For authentication, a REST authentication service is expected at the Authentication URL.

The following response from the REST authentication service indicates that user authentication is successful and that API Connect will use `cn=spoon,o=eatery` as the user identity.

```

HTTP/1.1 200 OK
Server: example.org
X-API-Authenticated-Credential: cn=spoon,o=eatery

```

For information on how to configure a User Security policy in an API assembly for use with an Authentication URL user registry, see [User Security policy](#).

For an example of an OAuth provider configuration that uses an Authentication URL user registry, see [Example - using multiple OAuth policies in an OAuth provider assembly](#).

API Connect considers any non-200 HTTP response code a failed user authentication attempt.

When an Authentication URL user registry is invoked, two HTTP response headers are available that include metadata in the access token or the response payload that contains the access token. For more information, see [OAuth external URL and authentication URL](#). The two metadata response headers are:

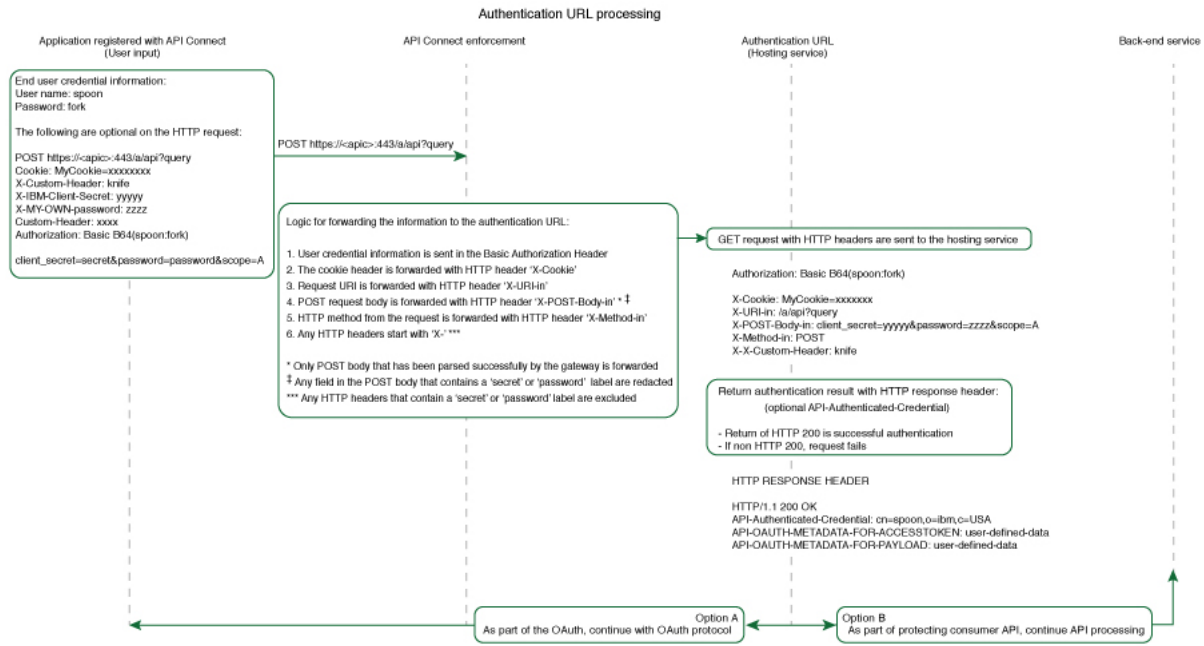
```

API-OAUTH-METADATA-FOR-ACCESSTOKEN
API-OAUTH-METADATA-FOR-PAYLOAD

```

When an Authentication URL is invoked, an HTTP response header is available to override the requested `scope` from the application. For more information, see [Scope](#). The response header is:

```
x-selected-scope
```



If you are using the DataPower® API Gateway rather than the DataPower Gateway (v5 compatible), this diagram is provided for guidance only and is not fully accurate for this release.

## Custom forms for user security

You can create custom forms for the authorization and identity extraction phase of OAuth.

### About this task

The Native OAuth provider configuration provides the capability for requiring additional authentication and authorization steps for user security. Custom HTML forms may be created to extract the identity of the user and to authorize the user. This capability applies to Implicit, Resource owner password, and Access code grant types.

- [Creating a custom HTML login form for user security](#)  
Custom HTML forms can be created for user security during the identity extraction stage in OAuth.
- [Creating a custom HTML authorization form for user security](#)  
Custom HTML forms can be created for user security during the authorization stage in OAuth.

## Creating a custom HTML login form for user security

Custom HTML forms can be created for user security during the identity extraction stage in OAuth.

### Before you begin

The Native OAuth provider configuration includes Identity Extraction when using the Implicit, Access code, or Resource owner password grant types. You have the option to select how to extract the user credential and one of the choices is Custom HTML Form. For more information, see [Configuring a native OAuth provider](#) when using API Manager or [Configuring a native OAuth provider](#) when using Cloud Manager. For more information, see [Configuring a native OAuth provider](#). This topic describes how to create the Custom HTML form for identity extraction.

### About this task

During three-legged OAuth definitions (Implicit flow, Resource owner password flow, and Access (Authorization) code flow, the user is presented with a form for signing in to the service provided by the API. You can present a custom form or a default form. Your custom form must fulfill certain requirements.

Important: The fields used by IBM® API Connect to inject information into your form have case-sensitive field names.

### Procedure

To create a custom sign-in form for your Native OAuth provider, complete the following steps:

1. Create a well formed XHTML document that will be parsed and transformed by API Connect to inject hidden fields.
2. For your XHTML form, set the method as `POST`, the encoding type as `application/x-www-form-urlencoded`, and the action as `authorize`. Add any other parameters that you require.  
For example:

```
<form method="POST" enctype="application/x-www-form-urlencoded" action="authorize">
```

3. Create a text input field that is named `username` and create a password input field named `password`.
4. Add the line `<EI-INJECT-HIDDEN-INPUT-FIELDS>`. This third element is a placeholder that API Connect replaces with input fields to complement the user-submitted data.
5. Create a button to initiate the sign-in process.  
For example:

```
<button
id=" home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.apionprem.doc_oauth_custom_login_form_login_b
utton" type="submit" name="login" value="true">Log in</button>
```

6. Optional: Add text that is displayed the first time that the user visits the sign-in page. Use the tag `<EI-LOGINFIRSTTIME>` for the text that you want to display.
7. Optional: Add text that appears when the user is returned to the sign-in page if they fail to authenticate. Use the tag `<EI-LOGINFAILED>` for the text that you want to display.
8. Optional: Have an error message displayed when an error in the custom form prevents it from being displayed to the user correctly. Use the tag `<EI-INTERNAL-CUSTOM-FORM-ERROR/>`; the message text is generated automatically. You should detect such errors during testing to prevent this error message being displayed to the end user.
9. Optional: You can add elements that are loaded from external sources, such as images or JavaScript.
10. Insert spacing and other features as you require. Completing Steps 1 through to 8 results in a form similar to the following example:

```
<html lang="en" xml:lang="en">
<head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/></head>
<body>
 <form method="POST" enctype="application/x-www-form-urlencoded" action="authorize">
 <h1>Please sign in</h1>
 <p>Username </p>
 <p ><input type="text" name="username" required="required" /> </p>
 <p>Password </p>
 <p ><input type="password" name="password" required="required" /> </p>
 <EI-INJECT-HIDDEN-INPUT-FIELDS/>
 <p > <button
id=" home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.apionprem.doc_oauth_custom_login_form_login_b
utton" type="submit" name="login" value="true">Log in</button> </p>

 <EI-LOGINFIRSTTIME>
 <p>If you have forgotten your user name or password, contact your system administrator.</p>
 </EI-LOGINFIRSTTIME>

 <EI-LOGINFAILED>
 <p >At least one of your entries does not match our records.
 If you have forgotten your user name or password, contact your system administrator.</p>
 </EI-LOGINFAILED>

 <EI-INTERNAL-CUSTOM-FORM-ERROR/>

 </form>
</body>
</html>
```

11. Make your form available at a URL of your choice.
12. If you have not already done so, configure your Native OAuth provider to use a Custom HTML form for identity extraction and provide the URL at which your form is available. For more information, see [Configuring a native OAuth provider](#) when using API Manager or [Configuring a native OAuth provider](#) when using Cloud Manager.

## Related tasks

---

- [Creating a custom HTML authorization form for user security](#)

## Related information

---

- [Configuring API security](#)

# Creating a custom HTML authorization form for user security

Custom HTML forms can be created for user security during the authorization stage in OAuth.

## Before you begin

---

The Native OAuth provider configuration includes user authorization when using the Implicit, Access code, or Resource owner password grant types. You have the option to select how to authorize application users and one of the choices is Custom HTML Form. For more information, see [Configuring a native OAuth provider](#) when using API Manager or [Configuring a native OAuth provider](#) when using Cloud Manager. This topic describes how to create the Custom HTML form for authorization.

## About this task

---

During three-legged OAuth definitions (Implicit flow, Resource owner password flow, and Access (Authorization) code flow, the user is presented with a form through which they grant permission to an application to access their data through the API on their behalf. You can present a custom form or a default form. Your custom form must fulfill certain requirements.

Important: The fields used by IBM® API Connect to inject information into your form have case-sensitive field names.

## Procedure

---

To create a custom authorization form for your Native OAuth provider, complete the following steps:

1. Create a well-formed XHTML document. This will be parsed and transformed by API Connect to inject hidden fields.
2. For your XHTML form, set the method as POST, the encoding type as `application/x-www-form-urlencoded`, and the action as `authorize`. Add any other parameters that you require.  
For example:

```
<form method="POST" enctype="application/x-www-form-urlencoded" action="authorize">
```

3. Add the line `<AZ-INJECT-HIDDEN-INPUT-FIELDS/>`. This line is a placeholder that API Connect will replace with input fields necessary for the completion of the OAuth process.
4. Create two buttons with the following code so that the user can grant or deny permission. Edit the text to suit your preferences.

```
<button class="cancel" type="submit" name="approve" value="false">No Thanks</button>
<button class="submit" type="submit" name="approve" value="true">Allow Access</button>
```

5. Optional: Display an error message when an error in the custom form prevents it from being displayed to the user correctly. Use the tag `<AZ-INTERNAL-CUSTOM-FORM-ERROR/>`; the message text is generated automatically. You should detect such errors during testing to prevent this error message being displayed to the end user.
6. Optional: You can add to the form HTML elements that will load features from external sources, such as images or JavaScript.  
For example, `<script src="http://www.example.com/example.js" />`
7. Insert spacing and additional elements as you require. Completing Steps 1 through to 6 results in a form similar to the following example:

```
<html lang="en" xml:lang="en">
 <head><title>Request for permission</title></head>
 <body class="customconsent">
 <div>
 <div>
 <form method="post" enctype="application/x-www-form-urlencoded" action="authorize">
 <AZ-INJECT-HIDDEN-INPUT-FIELDS/>
 <p>Greeting..</p><DISPLAY-RESOURCE-OWNER/>
 <p>This app </p><OAUTH-APPLICATION-NAME/><p> would like to access your data.</p>
 <div>
 <button class="cancel" type="submit" name="approve" value="false">No Thanks</button>
 <button class="submit" type="submit" name="approve" value="true">Allow Access</button>
 </div>
 </form>
 </div>
 </div>
 <AZ-INTERNAL-CUSTOM-FORM-ERROR/>
 </body>
</html>
```

8. Make your form available at a URL of your choice.
9. If you have not already done so, configure your Native OAuth provider to use a Custom HTML Form for authorization for User Security. Provide the URL as the endpoint at which your form is available. For more information, see [Configuring a native OAuth provider](#) when using API Manager or [Configuring a native OAuth provider](#) when using Cloud Manager.

## Related tasks

---

- [Creating a custom HTML login form for user security](#)

## Related information

---

- [Configuring API security](#)

## Securing an API with a JSON Web Token

There are two methods to secure your API with a JSON Web Token. You can use the `jwt-generate` command, or you can use a token that has been generated external to IBM® API Connect.

## About this task

---

JSON Web Token (JWT) is an OAuth 2.0 compliant method of authentication that can be useful to secure your API in API Connect.

You can secure your API with a JSON Web Token by using either of the following methods:

- Generate a token through the `jwt-generate` command, and then augment the response payload with your generated token replacing the `id` token.
- Use a token that was generated outside of API Connect and include it into the response payload, by using the metadata URL.

## Procedure

---

Create a `jwt-generate` policy in the assembly.

- a. In API Manager, open the Assembly tab.
- b. Add a `jwt-generate` policy to the assembly for the API.

## Troubleshooting OAuth

You can find the answers to some common questions in the Developer Portal, or in support communities and forums in DeveloperWorks, on GitHub, or on YouTube.

You can use the following links to go to the topics:

- [OAuth 2.0 support in the developer portal test tool](#)
- [OAuth requires quorum in a multi-node cluster](#)

## OAuth 2.0 support in the developer portal test tool

---

OAuth 2.0 support is exposed as an API through the provider OpenAPI definition. You can use the test tool that is included with API Connect to test the OAuth 2.0 configuration. For more information, see [Testing an API using the Developer Portal test tool](#).

## OAuth requires quorum in a multi-node cluster


---

In a multi-node cluster, OAuth operations will fail if quorum is lost. Quorum requires that the number of active nodes is greater than 50% of the total number of nodes in the cluster.

---

## Configuring the cloud settings

Before you can add a provider organization to your cloud, you must specify your cloud settings to configure an email server for notifications as well as user registry options and catalog defaults.

The cloud settings let you define basic configuration for your deployment. To view the current settings, log in to Cloud Manager and click  in the navigation pane.

Tip: When you are familiar with the cloud settings, you can modify them using the [Toolkit command-line interface](#) instead of the user interface.

Visit the following pages to configure settings:

### Overview

Provide a title and description for your cloud administrator organization. For more information, see [Change the name of your cloud](#).

### Onboarding

Specify time-out settings for onboarding invitations, password resets, access tokens, and refresh tokens. To specify a timeout value, select or type an integer value in the Number field and then choose a unit of time (Seconds, Minutes, or Hours) in the Unit field.

For more information on setting timeouts, see the following topics:

- [Configuring invitation timeouts](#)
- [Configuring the password-reset notification and timeout](#)
- [Configuring timeouts for access tokens and refresh tokens](#)

### User Registries

Configure one or more user registries to authenticate users of Cloud Manager or API Manager. The user credentials for all users who login to each applications must be stored in the specified registries. For more information on configuring user registries, see [Selecting user registries for Cloud Manager and API Manager](#).

### Roles

Create roles and assign them to members of the Admin organization to specify which tasks members are authorized to perform. For more information on assigning roles, see [Administering members and roles](#).

### Role Defaults

The Role Defaults for the Admin organization are not configurable, but you can customize the base roles for provider organizations and consumer organizations. For more information on customizing base roles, see [Role Defaults overview](#).

### Endpoints

View the list of endpoints configured for API Manager, provider APIs, and consumer APIs during the installation process.

### Notifications

Select the email server for your on-premises cloud notifications, configure notification settings, and specify the sender name and email address to be included in the emails. You can also preview and customize email templates used for notifications. For more information on setting up notifications, see [Setting up notifications](#).

Note: Be sure to [configure the email server](#), because it is needed for sending invitations and other notifications to users.

### Catalog Defaults

Specify one or more gateway services to be provided to every Catalog by default, so that Products (which contain APIs) can be made available to users. When a new Catalog is created in a provider organization, it is assigned to the default gateway service. Any changes you make to these settings do not affect existing Catalogs.

For more information on selecting gateway services, see [Configuring the default gateway services for Catalogs](#).

### Audit Setting

Configure auditing settings to monitor API calls and log information about the calling users, the time of each call, and the activity involved in each event. The audit focuses on "who" did "what" and "when", including the request payload and the result of the operation. All create, update, and delete operations are audited.

Generally, read operations are only audited if the calling user logged in with an OIDC Provider user registry. For more information about configuring the use of auditing, see [Configuring audit logging to monitor user operations](#).

Use the following list to locate instructions for using specific cloud settings:

- [Change the name of your cloud](#)  
You can change the name of your cloud.
- [Configuring an email server for notifications](#)  
Connect to an email server in your API Connect on-premises cloud. The email server sends registration invitations and other event-driven emails. You must configure the email server before you add any provider organizations or register a Portal service. Otherwise, the owner of the organization will not receive the email with the details that they require to access the API Manager.
- [Setting up notifications](#)  
Select an email server and set up the "sent from" information for notifications.
- [Configuring invitation timeouts](#)  
You can set the time period for when registration invitations for Cloud Settings access and the Admin organization expire.
- [Configuring API Key timeouts](#)  
You can set the time period for when an API Key expires.

- [Using the CLI to modify cloud settings](#)  
Use the toolkit CLI to edit the cloud settings.
- [Allowing the API key to be used for multiple logins](#)  
Use the toolkit CLI to enable API keys to be used for multiple toolkit logins.
- [Configuring Base64 encoding for temporary tokens](#)  
How to enable and disable Base64 encoding for temporary tokens.
- [Configuring the password-reset notification and timeout](#)  
Configure the time allowed for a user to reset a password, and customize the notification that the user receives.
- [Selecting user registries for Cloud Manager and API Manager](#)  
Select the user registries to authenticate users for Cloud Manager and API Manager.
- [Configuring timeouts for access tokens and refresh tokens](#)  
Configure timeout values for the access tokens and refresh tokens for user registries.
- [Viewing platform and UI endpoints](#)  
A view-only list of platform and UI endpoints is provided.
- [Configuring the default gateway services for Catalogs](#)  
Configure the list of default gateway services that are available to new Catalogs.
- [Managing the public/private key pairs for signing tokens](#)  
Manage the public/private key pairs that are used for signing tokens in API Connects.

---

## Change the name of your cloud

You can change the name of your cloud.

### About this task

---

You can change the name of your cloud using the Overview page.


One of the following roles is required to change the name of your cloud:

- Administrator
- Owner
- A custom role with the `cloud settings:manage`

### Procedure

---

Follow these steps to change the name of your cloud:

1. In the Cloud Manager, click  Settings.
2. Select Overview..
3. Click Edit to change the name of your cloud.
4. Enter a Title and the name is auto-generated. Enter a brief Summary and click Save.

---

## Configuring an email server for notifications

Connect to an email server in your API Connect on-premises cloud. The email server sends registration invitations and other event-driven emails. You must configure the email server before you add any provider organizations or register a Portal service. Otherwise, the owner of the organization will not receive the email with the details that they require to access the API Manager.

### Before you begin

---

You must complete the tasks in the following section:

1. [Defining your Cloud Manager topology.](#)

### About this task

---

This task can be completed by users who are assigned one of the following roles:

- Cloud Owner
- Cloud Administrator
- Topology Administrator
- Custom role with `Settings:Manage` permission


The email server sends invitation emails with activation links and other system messages. More than one email server may be configured in Cloud Manager, but only one email server may be selected as the active server to send notifications at any one time. For instructions on how to select the active email server and to view a list of the email messages that are sent, see [Setting up notifications.](#)

Note: If you do not select an email server for sending notification emails, it is not possible for a user to reset their password from either the Cloud Manager or API Manager login pages.

### Procedure

---

To configure one or more email servers to use for email notifications in Cloud Manager, complete the following steps:

1. In the Cloud Manager, click  Resources.
2. Choose Notifications.
3. Enter the values to configure the email server.

Field	Description
Title (required)	Enter a descriptive title for the email server. This title will display on the screen.
Name (required)	This field is auto-populated by the system and used as the internal field name.
Address (required)	IP address for the email server
Port (required)	Port that the email server runs on
Authenticate User	Enter a user name to use for authorization with the SMTP server. The user name is required if the SMTP server requires authentication.
Authenticate Password	Enter the password for the Authenticate User. This password is used for authorization with the SMTP server. The password is required if the SMTP server requires authentication.
TLS Client Profile (optional)	Select an optional TLS Client Profile that will be used to communicate with the email server if the Secure Connection option is selected.
Secure Connection	Select this option to specify that TLS security is used. The following conditions apply: <ul style="list-style-type: none"> <li>• If this option is not selected then TLS security is not used even if the email server is configured to use the STARTTLS protocol.</li> <li>• If this option is selected and a TLS Client Profile is selected, that profile is used and the STARTTLS protocol configuration on the email server is honored.</li> <li>• If this option is selected and no TLS Client Profile is selected, the Default TLS client profile is used and the STARTTLS protocol configuration on the email server is honored.</li> </ul>

4. You can send a test email to confirm that the email server is properly configured. Click Test email in the Test Connection panel to open a dialog window. Enter one or more valid email addresses in the Recipients field. Separate multiple recipients with commas. Click Send test email. Check that the test email was received by the recipients.
5. If the test email is successful, click Save to save the email server configuration. If the test is unsuccessful, check the configuration values and update as needed.  
Note: In order to send emails, you must select an email server as your active server in Settings > Notifications.

## Results

The email server is configured.

## What to do next

You must now select the email server to use as your active email server in Cloud Settings > Notifications. You can enter the sender information, and customize the email templates. For more information, see [Setting up notifications](#).  
Note: If you do not select an email server for sending notification emails, it is not possible for a user to reset their password from either the Cloud Manager or API Manager login pages.

## Setting up notifications

Select an email server and set up the "sent from" information for notifications.

## Before you begin

You must configure an email server to use for your notifications in Resources > Notifications, before you can set up the notification configuration in Settings > Notifications. See [Configuring an email server for notifications](#) for more information.

## About this task

In Settings > Notifications, you select the email server that you want to use as the active server to send notification emails. An active email server is required to send email notifications. More than one email server may be configured in Cloud Manager, but only one email server may be selected to send notifications at any one time. The email server must be configured and selected for notifications before adding a Portal Service to your cloud.


You also configure the sender information to be included on all the emails and optionally, edit the standard text for the email templates.

One of the following roles is required to add roles to select the email server, enter the sender information, and edit the email templates:

- Administrator
- Owner
- A custom role with the `Cloud settings:Manage` permission

Note: If you do not select an email server for sending notification emails, it is not possible for a user to reset their password from either the Cloud Manager or API Manager login pages.

## Procedure

1. In the Cloud Manager, click  Settings.
2. In the Settings navigation list, click Notifications.
3. To configure the active email server:
  - a. Click Edit in the Sender & Email Server section.
  - b. Select an email server in the list; a check mark is displayed next to the currently selected server.



- c. Click Save.
4. To delete an email server from the list, click the options menu next to the server name and select Delete.
5. To change the sender name and email address for notifications:
  - a. Click Edit in the Sender & Email Server section.
  - b. Fill in the Name and Email address that notifications will be sent from.
  - c. Click Save.
6. To preview and edit the notification email templates, see [Customizing email notification templates](#)

## What to do next

You can now [create a provider organization](#).

- **Customizing email notification templates**  
API Connect includes a set of email templates for invitations and other notification emails. You can preview the templates and customize the text if required. You can also style the notifications to match your business theme.

## Customizing email notification templates

API Connect includes a set of email templates for invitations and other notification emails. You can preview the templates and customize the text if required. You can also style the notifications to match your business theme.

### About this task

The API Connect emails are sent automatically when certain system events occur. The email templates are organized by scope, as explained in Table 1.

Table 1. Notification scopes

Scope	Events that trigger the notification
admin	Admin organization activities such as invitations, portal sign up, and password reset.
catalog	Activities related to Catalogs; for example: such as application life cycle events, invitations to catalogs, product approvals, and subscription approvals.
consumer	Activities to consumer applications and subscription requests in the Developer Portal, invitations to Consumer organizations, and password reset requests for Developer Portal accounts.
provider	Invitations
space	Invitations

Email notifications contain a sender name and address that's based on a hierarchical search of the configured sender details in API Connect. Depending on the email template being sent, the search can start at the Space level, and then go through the Catalog, Provider Organization, and the Cloud Manager levels. The sender details that are used will be the first set of configured details found during this hierarchical search. To view which search path API Connect takes when looking for the sender details for each notification template, see [Configuring sender details for email notifications](#) in the Managing your APIs section.


One of the following roles is required to configure email notifications:

- Administrator
- Owner
- A custom role with the `Cloud settings:Manage` permission

Note: You can also customize notifications by using the developer toolkit CLI, or by using the API Connect REST APIs. See [Using the CLI to modify cloud settings](#), or [API Connect REST APIs](#), for more information.

If you want to configure the notifications at the Provider Organization level, see [Configuring notifications](#) in the Managing your APIs section.

## Procedure

1. In the Cloud Manager, click  Settings.
2. In the Settings navigation list, click Notifications.  
On the Notifications page, templates display in a list, sorted by scope. When selecting a template to modify, be sure to select the version that applies to the appropriate scope.

3. To preview the text for a template, select Preview from the actions menu next to the template name.
4. To edit the text for the template, either click the template name, or select Edit in the action menu.  
When a template is opened, an attempt is made to get the language from the browser setting, but if this isn't possible the template defaults to English.

The text includes variables; for example, `{{ catalog }}`. The notification text is based on Handlebars syntax. Most variables are enclosed in double curly braces `{{ }}`, but can be enclosed in triple curly braces `{{{ }}} to disable HTML escaping, when the variable is a URL link for example. For more information on Handlebars, see https://handlebarsjs.com/.`

To obtain the complete list of variables that are available for a particular notification template, complete the following steps:

- a. Log in to the management server from the command line as a member of the cloud administration organization; for details, see [Logging in to a management server](#). You can use the same management server URL, user name, and password in the login command that you use to log in to the Cloud Manager user interface.
- b. Enter the following command:

```
apic notification-templates:get template_name --server mgmt_endpoint_url --scope cloud --subcollection template_scope
--fields variables --output -
```

where:

- *template\_name* is the name of the required notification template, as displayed in the Template column in the user interface.
- *template\_scope* is the scope name displayed in the Scope column alongside that template.

For example:

```
apic notification-templates:get member-invitation --server https://myserver.com --scope cloud --subcollection catalog
--fields variables --output -
```

The variables that are available for the template are displayed, for example:

```
variables:
- org
- catalog
- activationLink
- expiresAt
- originator
- originatorFirstName
- originatorLastName
- originatorEmail
- username
- email
- firstName
- lastName
```

The `--output -` parameter causes the command output to be written to the command line. You can specify `--output filepath` to have the output written to a .yaml file at the specified location, or omit it altogether to have a file written to the current folder.

- To view and edit the notification template in a different language, select one of the following supported languages from the View template in drop-down list:
  - Chinese (Simplified)
  - Chinese (Traditional)
  - Czech
  - Dutch
  - English (US English)
  - French
  - German
  - Italian
  - Japanese
  - Korean
  - Polish
  - Portuguese
  - Russian
  - Spanish
  - Turkish
- Edit the Subject as required.
- Select the Content type that you want to use for the template, from HTML, PlainText, or Both. The default content type is PlainText. An edit window for the selected content type is displayed, or both edit windows are displayed if Both is selected.
- Edit the body of the template as required.

For HTML content, only the tags and their attributes that are shown in the following table are allowed.

Table 2. List of allowed HTML tags and their attributes

HTML tag	Attribute
<a>	"class", "href", "hreflang", "style"
<em>	"class", "style"
<strong>	"class", "style"
<cite>	"class", "style"
<blockquote>	"class", "cite", "style"
<code>	"class", "style"
<ul>	"class", "type", "style"
<ol>	"class", "start", "type", "style"
<li>	"class", "style"
<dl>	"class", "style"
<dt>	"class", "style"
<dd>	"class", "style"
<h1>	"class", "id", "style"
<h2>	"class", "id", "style"
<h3>	"class", "id", "style"
<h4>	"class", "id", "style"
<h5>	"class", "id", "style"
<h6>	"class", "id", "style"
<p>	"class", "style"
<div>	"class", "style"
 	"class", "style"
<span>	"class", "style"
<img>	"class", "src", "alt", "data-entity-type", "data-entity-uuid", "data-align", "data-caption", "width", "height", "style"
<table>	"class", "id", "style"
<tr>	"class", "id", "style"
<td>	"class", "id", "style"

If an HTML tag that is not allowed is used in a notification, the tag and its contents are displayed in the email as plain text.

Images can be used by adding a `<img`

`src="https://path/to/image.png"/>` tag in the template. The `src` attribute for the image must be a fully qualified web URL, and must be externally accessible so that the email recipients can access the image. It's not possible to reference local images, they must be fully qualified URLs. It's also not possible to embed or attach images or other files in the emails.

- Click Save when done.
  - Note: Edits made to a template are saved only for the specific language version that is edited.

## Related information

- [Configuring notifications](#)

---

## Configuring invitation timeouts

You can set the time period for when registration invitations for Cloud Settings access and the Admin organization expire.

---

### About this task


One of the following roles is required to change the name of your cloud:

- Administrator
- Owner
- A custom role with the `cloud settings:manage` permission

---

### Procedure

Follow these steps to set the invitation timeout:

1. In the Cloud Manager, click  Settings.
2. In the Settings navigation list, click Onboarding.
3. Click Edit next to the invitation timeout setting that you want to change:
  - Cloud Settings Invitation Timeout
  - Admin Organization Invitation Timeout
4. To specify a timeout value, select or type an integer value in the Number field and then choose a unit of time (Seconds, Minutes, or Hours) in the Unit field.
5. Click Save.

---

### Results

Invitations contain a link, which expires after the specified the timeout. If the user does not register before the expiration, you can send a new invitation.

---

## Configuring API Key timeouts

You can set the time period for when an API Key expires.

---

### About this task


One of the following roles is required to change the name of your cloud:

- Administrator
- Owner
- A custom role with the `cloud settings:manage` permission

---

### Procedure

Follow these steps to set the API Key timeout:

1. In the Cloud Manager, click  Settings.
2. In the Settings navigation list, click Onboarding.
3. Click Edit next to the API Key timeout setting.
4. To specify a timeout value, select or type an integer value in the Number field and then choose a unit of time (Seconds, Minutes, or Hours) in the Unit field.
5. Click Save.

---

### Results

API Key times out based on the set timeout value.

---

## Using the CLI to modify cloud settings

Use the toolkit CLI to edit the cloud settings.

---

### About this task

Some cloud settings can be modified in the Cloud Manager UI, but all settings can be modified with the toolkit CLI. You can modify the cloud settings configuration file and upload the changes, or issue commands to modify individual settings directly. If multiple people modify the cloud settings, any newer settings overwrite older settings regardless of the method used to update the settings.

For more information about working with the toolkit CLI, see the following topics:

- For details on using **apic** to modify cloud settings, see [apic cloud-settings:update](#).
- For details on all the cloud settings commands, see [apic cloud-settings](#)

## Procedure

1. Run the following command to log in to a management server as a member of the cloud administration organization:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm admin/identity_provider
```

Cloud settings are stored on a management server and you must log in to the server before you can access the settings with the CLI.

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`.

For more information on logging in, see [Logging in to a management server](#).

2. Optionally retrieve the current list of cloud settings with the following command:

```
apic cloud-settings:get --server mgmt_endpoint_url --output -
```

where `mgmt_endpoint_url` is the URL of the server where the cloud settings are stored.

3. Add or modify settings using one of the following methods:

- Issue each setting from the command line.  
For simple changes, you can issue commands to modify settings without editing the configuration file.

- Issue the following command to start the update process:

```
apic cloud-settings:update --server mgmt_endpoint_url -
```

where `mgmt_endpoint_url` is the URL of the server where the cloud settings are stored. Include the hyphen at the end of the command to indicate that input follows on the next command line.

- Type the information for a single setting on one line, and press ENTER to submit it.  
Specify each setting on a single line and use the following format to specify the setting's name and value:

```
setting_name: setting_value
```

For example, the following command updates a single setting:

```
apic cloud-settings:update --server mgmt_endpoint_url -
refresh_expires_in: expiration time
Ctrl+D
```

- Type Ctrl+D to indicate that the data entry is complete, as shown in the example.
- Upload a file containing the settings.  
If you have multiple settings or lengthy values to add or modify, you might prefer to make all of the changes at once by creating or modifying the configuration file and then uploading it.

- Create a `.yaml` file with your settings.  
You can choose the file name but must use YAML format. The file can contain as few or as many settings as needed.

Settings use the following format; notice that the setting's name must be enclosed in quotation marks as shown in the example. The last setting in the file does not end with a comma.

```
"setting_name": setting_value,
```

- Upload the file with the following command:

```
apic cloud-settings:update --server mgmt_endpoint_url my_cloud_settings.yaml
```

For information on using input files with the toolkit CLI, see [./com.ibm.apic.toolkit.doc/rapic\\_cli\\_command\\_line\\_input.html](#).

4. When you are ready to log out of the management server, use the following command:

```
apic logout --server mgmt_endpoint_url
```

## Results

The settings are cached for a maximum of five minutes, so you might experience a delay before your changes take effect.

---

## Allowing the API key to be used for multiple logins

Use the toolkit CLI to enable API keys to be used for multiple toolkit logins.

### About this task

---

Users can log in to the API Connect management server with the toolkit using an API key. By default, the API key can be used only once. You can change the behavior by enabling the API key to be used multiple times.

Use the CLI to set the `api_key_multiple_uses` setting to `true` and enable the API key to be used multiple times for logins. By default, the setting is set to `false` and is disabled. For information on configuring Cloud Manager settings with the CLI, see [Using the CLI to modify cloud settings](#).

---

## Configuring Base64 encoding for temporary tokens

How to enable and disable Base64 encoding for temporary tokens.

### About this task

---

You can enable or disable Base64 encoding for temporary tokens. Temporary tokens are invitations, and password reset tokens. When Base64 encoding is enabled, the temporary token is encoded in Base64 format. When Base64 encoding is disabled, the temporary token remains in the JSON Web Token (JWT) format, and means that the length of the URL that the user receives is reduced.

Base64 encoding is enabled by default.


One of the following roles is required to configure Base64 encoding:

- Administrator
- Owner
- A custom role with the `cloud_settings:manage` permission

### Procedure

---

Follow these steps to configure Base64 encoding for temporary tokens:

1. In the Cloud Manager, click  Settings.
2. In the Settings navigation list, click Onboarding.
3. Click Edit next to the Base64 encoding setting.
4. Select Enable to enable Base64 encoding, or Disable to keep temporary tokens in the JSON Web Token (JWT) format.
5. Click Save.

### Results

---

You have successfully enabled or disabled Base64 encoding for temporary tokens.

---

## Configuring the password-reset notification and timeout

Configure the time allowed for a user to reset a password, and customize the notification that the user receives.

### Before you begin

---

Configure an email server that can be used for mailing notifications to users. If you do not configure an email server, a link cannot be sent to a user who wants to reset a password. For information on configuring the email server, see [Setting up notifications](#).


### About this task

---

When a user clicks Forgot Password? and provides an email address on the sign-in page, an email is generated with a link where the user can reset the password. When you deploy API Connect, the password-reset settings are configured with default values. You can optionally modify the time-out setting to set a new expiration for resets as well as the notification that is emailed to users.

### Procedure

---

1. In the Cloud Manager, click  Settings.
2. In the Settings navigation list, click Onboarding.
3. Configure the password reset timeout:
  - a. In the Settings navigation list, click Onboarding.
  - b. Click Edit next to Reset Password Time to Live.
  - c. To specify a timeout value, select or type an integer value in the Number field and then choose a unit of time (Seconds, Minutes, or Hours) in the Unit field.
  - d. Click Save.

4. Customize the notification template for password resets:
  - a. In the Settings navigation list, click Notifications.
  - b. In the Notifications Templates list, select the appropriate version of the "password-reset" template by clicking the name. You can create different versions of the template for different user roles.
  - c. Customize the template as explained in [Customizing email notification templates](#).
  - d. Click Save.

---

## Results

Password reset notifications contain a link, which expires after the specified the timeout. If the user does not reset the password before the link expires, they can request a new link.

---

## Selecting user registries for Cloud Manager and API Manager

Select the user registries to authenticate users for Cloud Manager and API Manager.

---

### About this task

In Settings > User registries, you select one or more user registries to authenticate users of Cloud Manager and API Manager. The user credentials for all users who login to the applications must be stored in the selected registries.

One of the following roles is required to select the user registries for Cloud Manager and API Manager:


- Administrator
- Owner
- A custom role with the `cloud settings:manage` permission

For details on how to control which user registries can be used by provider organizations for authenticating Developer Portal users, or for securing APIs, see [Setting visibility for a user registry](#).

---

### Procedure

To select the user registries for Cloud Manager and API Manager, follow these steps:

1. In the Cloud Manager, click  Settings.
2. In the Settings navigation list, click User Registries.
3. The current list of user registries is displayed for Cloud Manager and for API Manager. To add user registries, click Edit in either the Cloud Manager or API Manager panel.  
Note: The list of user registries is derived from the user registries that were previously configured as a Resource in Resources > User Registries. For more information, see [User registries overview](#).
4. Select the user registry by placing a check mark next to the name in the list. Remove a user registry by removing the check mark.
5. Click Save.

---

### Results

The user registry list is updated to reflect the selected user registries available for authenticating users.

---

### What to do next

To set a user registry as the default, click the options menu next to the registry name in the list, and select Set as default.

- [Setting a default user registry for Cloud Manager and API Manager](#)  
You can select a default user registry for Cloud Manager and and for API Manager.

---

### Related tasks

- [Configuring an LDAP user registry in the Cloud Manager](#)
- [Configuring an Authentication URL user registry](#)
- [Configuring a Local User Registry](#)

---

## Setting a default user registry for Cloud Manager and API Manager

You can select a default user registry for Cloud Manager and and for API Manager.

---

### About this task

In Settings > User registries, you can set a user registry as the default user registry.


One of the following roles is required to select the user registries for Cloud Manager and API Manager:

- Administrator

- Owner
- A custom role with the `cloud_settings:manage` permission

## Procedure

---

1. In the Cloud Manager, click  Settings.
2. In the Settings navigation list, click User Registries.
3. To set a user registry as the default, click the options menu next to the registry name in the list, and select Set as default.

Note: The list of user registries is derived from the user registries that were previously configured as resources in Resources > User Registries. For more information, see [User registries overview](#)

## Results

---

The user registry list is updated to reflect the default user registry.

## Related tasks

---

- [Configuring an LDAP user registry in the Cloud Manager](#)
- [Configuring an Authentication URL user registry](#)
- [Configuring a Local User Registry](#)

# Configuring timeouts for access tokens and refresh tokens

---

Configure timeout values for the access tokens and refresh tokens for user registries.

## About this task

---

IBM® API Connect supports the use of access tokens and refresh tokens for user registries that authenticate users of the Cloud Manager, the API Manager, and the Developer Portal.

Access tokens allow users to sign in and remain logged in for a specified amount of time (the timeout period). Refresh tokens allow the access token to be renewed automatically without requiring the user to sign in again. Access tokens are configured by default, and you can modify the timeout setting. Refresh tokens are disabled by default but are configured with a default timeout when you enable them.

Refresh tokens, when enabled, are generated every time an access token is created. Refresh tokens can be used to obtain a new access token without having to prompt a user to re-login. If you enable the use of refresh tokens, then when the token expires it is refreshed automatically in the background so that the user can continue working without interruption. When the user logs out, the refresh token is revoked and the application can no longer use it.

Note: If users log in with non-OIDC user registries, the use of refresh tokens is supported in API Manager, API Designer, the Developer Portal, and the CLI Toolkit. When the refresh token expires, the user is always redirected to the login page.

If users log in with OIDC user registries:

- The use of refresh tokens is supported only in the Developer Portal and the CLI Toolkit
- If the OIDC providers do not support refresh tokens, API Connect will not issue a `refresh_token`, regardless of the setting of `refresh_token_enabled` in the cloud settings.
- For more information about OIDC configuration, see [Configuring an OIDC user registry](#).


You can configure access tokens and refresh tokens using the [Cloud Manager UI](#), or the [Toolkit CLI](#).

## Using the UI to configure tokens

---

Configure timeout settings for access tokens and refresh tokens in the Cloud Manager Settings pages.

### Procedure

1. Click  in the navigation pane.
2. Configure the timeout for access tokens:
  - a. On the Onboarding page, click Edit next to Access Token Time to Live.
  - b. To specify a timeout value, select or type an integer value in the Number field and then choose a unit of time (Seconds, Minutes, or Hours) in the Unit field.
  - c. Click Save.
3. Configure the timeout for refresh tokens:
  - a. On the Onboarding page, click Edit next to Refresh Token Time to Live.
  - b. To specify a timeout value, select or type an integer value in the Number field and then choose a unit of time (Seconds, Minutes, or Hours) in the Unit field.
  - c. Click Save.

## Using the CLI to configure tokens

---

Use the Toolkit CLI to configure timeout settings for access tokens and refresh tokens.

### About this task

Use the following properties to specify refresh tokens for cloud settings:

- `access_token_expires_in`  
Integer. Represents the expiration time, in seconds, for access tokens issued by API Connect.

- **refresh\_expires\_in**  
Integer. Represents the expiration time, in seconds, for refresh tokens issued by API Connect. Must be greater than **access\_token\_expires\_in**.
- **refresh\_token\_enabled**  
Boolean. Disabled by default. To enable, set to **true**. When enabled, generates a **refresh\_token** field in the response to the **/token** API call.

The following extract from a cloud settings file shows example settings for these properties:

```
"type": "cloud_setting",
"api_version": "2.0.0",
"name": "cloud-setting",
"access_token_expires_in": 28800,
.
.
.
"refresh_expires_in": 57600,
"refresh_token_enabled": false,
.
.
.
```

Use the toolkit CLI (**apic**) to update the cloud settings with new values for the properties. You can either specify values as command line arguments, or enter them manually in a configuration file as explained in [Using the CLI to modify cloud settings](#).

---

## Viewing platform and UI endpoints

A view-only list of platform and UI endpoints is provided.

### About this task

You can view the list of endpoints configured during the installation process. Included in this view-only list are the following endpoints:


- API Manager URL - the URL for opening the API Manager UI
- Platform REST API endpoint for the Admin and Provider REST APIs - this endpoint handles the REST APIs for Admin and Provider functions.
- Platform REST API endpoint for consumer REST APIs - this endpoint handles the REST APIs for consumer (Portal) functions.

One of the following roles is required to view endpoints:

- Administrator
- Owner
- A custom role with the `cloud settings:manage` or `cloud-settings: view permissions`

### Procedure

To view the endpoints, follow these steps:

1. In the Cloud Manager, click  Settings.
2. Select Endpoints.
3. View the list of endpoints.

---

## Configuring the default gateway services for Catalogs

Configure the list of default gateway services that are available to new Catalogs.

### About this task

One of the following roles is required to configure Catalog Defaults:

- Administrator
- Owner
- A custom role with the `cloud settings:manage` permission

You publish APIs by adding them to a Product and then publishing the Product to a Catalog. To be able to publish Products to a Catalog, the Catalog must be assigned at least one gateway service so that the APIs in the Product are available to be called at a gateway service endpoint. You can assign more than one gateway service to a Catalog, and they can be of mixed types: DataPower® API Gateway, DataPower Gateway (v5 compatible), and Event Gateway Service.


A Product is gateway type specific: DataPower API Gateway, DataPower Gateway (v5 compatible), or Event Gateway Service. By default, when you publish a Product to a Catalog, it is published to all the assigned gateway services of the same gateway type as the Product, but you can choose to publish the Product only to selected gateway services, of that type, at publish time; see [Publishing a draft Product](#) for more information. The APIs in the Product will be available to be called at all the specified gateway service endpoints.

The procedure described in this topic specifies the gateway services that are assigned by default to newly created Catalogs, but you can change the assigned gateway services for a specific Catalog; see [Creating and configuring Catalogs](#).

### Procedure



To configure the gateway services that are assigned to new Catalogs by default, complete the following steps:

1. In the Cloud Manager, click  Settings.
2. In the Settings navigation list, click Catalog Defaults.
3. Click Edit to change the list of default gateway services.
4. Select the required gateway services, then click Save.

## Results

---

When a new Catalog is created in a provider organization, the default gateway services are automatically assigned to it. Any changes you make to the list of default services affects only the Catalogs that are created after you save your changes. Existing Catalogs are not affected.

## Related information

---

- [Working with Catalogs](#)
- [API Connect gateway types](#)

---

## Managing the public/private key pairs for signing tokens

Manage the public/private key pairs that are used for signing tokens in API Connects.

API Connect uses public-key cryptography to secure tokens. By default, an API Connect deployment receives one public/private key pair, which is used for signing and verifying tokens. When a token is created, API Connect signs it with the deployment's private key. Any third-party can verify that digital signature with the deployment's public key.

The public/private key pair is generated for you by API Connect on the first start-up after installation, upgrade, or migration. The keys are stored in a keystore object.

## Why do you need public/private key pairs?

---

API Connect uses a public/private key pair to sign and verify the following tokens:

- `access_token`: Verifies that the user has permission to access resources.
- `id_token`: Stored within the access token to specifies the user's ID. **Confirm?**
- `temporary_token`: Provides for short-lived access so that the user can log in and receive an access token. For example, when the user receives a link for resetting a password or to register for an account.

## How does APIC use the public/private key pair?

---

Each API Connect deployment is assigned a public key, which is made openly available, plus a corresponding private key, which can only be accessed by that deployment. To verify a user's access to resources for a transaction, API Connect signs every token with the private key. Any other party can use the openly available public key to verify the signature. Because the private key is not shared, it can be used repeatedly instead of generating a new value for each signature.

When your app accesses an API endpoint, the `access_token` is in JWT format and includes a "kid" value that is the new key.

Note: The signature for public/private key schema is larger than for shared secrets, so the `access_token` that is returned will also be larger. If you encounter issues with headers becoming too large, refer to the following topic: [Resolving login problems by increasing HTTP header size](#).

## How can you manage public/private key pairs?

---

By default, your deployment is assigned one key pair, which is stored in a keystore and never expires. The same key pair is used for signing and verify all tokens. The key pair consists of a public key and a private key:

- `key_signing`: The private key that API Connect uses for signing tokens
- `key_verify`: The public key that anyone can access and use for verifying token signatures

API Connect uses the following cloud settings to determine which public/private keystores are used for each type of token:

- `access_token_keystore_urls`: [`keystore_url_0`, `keystore_url_n`]
- `id_token_keystore_urls`: [`keystore_url_0`, `keystore_url_n`]
- `temporary_token_keystore_urls`: [`keystore_url_0`, `keystore_url_n`]

Each setting specifies an array of URLs where the keystores are hosted, with the keystore at position 0 representing the current key pair. You can optionally generate additional key pairs, store them in new keystores, and use them with the tokens to enhance security. Each key pair is stored in its own keystore and can be set to expire at a different time. Providing a different key pair for each type of token is a good practice because it limits the damage if a token is compromised.

If you specify one keystore for a token, that key pair is used for every new token of the same type, and it never expires. If you specify multiple keystores, you can assign a lifetime to each and when the keys in the first keystore expire, the keys are obtained from the next keystore in the list, and so on.

You can use the public/private key pairs and keystores to perform the following operations with tokens:

Sign a token

The first keystore in the array is the current keystore; use its `key_signing` key to sign tokens.

Verify a token's signature

To verify a token's signature, start with the current keystore (the first in the array). If the token doesn't match the `key_verify` key, move to the older keystores and compare the token with the `key_verify` value in each.

Add key pairs

When you create a new key pair, add the keys to a separate keystore, and then add that keystore's URL to the end of the token's keystore array. The first keystore in the array is the "current" keystore, and subsequent keystores in the array are used in turn.

#### Revoke key pairs

If you suspect that a token is compromised, you can revoke the keys for signing and verifying it. To revoke a key pair, delete the corresponding keystore from the token's keystore array. Removing a keystore removes the keys that were used for signing and verifying tokens, and invalidates all of the tokens that were issued with that key pair.

Attention: You must always retain at least one key pair for signing and verifying tokens.

#### Rotate through keystores to periodically change the key pairs in use

To rotate through key pairs for a token, configure at least two keystores for the token. Specify the "current" keystore first in the array, followed by the "next" keystores. Make sure that the each keystore's lifetime overlaps that of the next keystore so that keys from the current keystore don't expire before the new key pair takes effect.

Before you configure the lifetime for a keystore, you can refer to the Onboarding section of the Cloud Settings page to quickly see the lifetime that is configured for each type of token. Then make sure you specify enough keystores, with long enough lifetimes, to span each token's lifetime. For example, if the `temporary_token` lives for 72 hours and the key used for signing it lives for 24 hours, you need to provide at least 3 sets of keys to span the token's lifetime:

```
temporary_token: [keystore1, keystore2, keystore3]
```

- **Configuring public/private key pairs for tokens**

Configure public/private key pairs for signing and verifying tokens in your API Connect deployment.

---

## Configuring public/private key pairs for tokens

Configure public/private key pairs for signing and verifying tokens in your API Connect deployment.

### About this task

By default, an API Connect deployment receives one public/private key pair, which is used repeatedly. The public/private key pair is generated for you the first time you start API Connect after installing or upgrading it. The keys are stored in a keystore object. You can add and revoke keys, and configure a rotation through keys for enhanced security.

You can configure key pairs for tokens in the [Cloud Manager user interface](#), or by [using the Toolkit CLI](#).

---



## Using the Cloud Manager UI to configure public/private key pairs

Use the Cloud Manager user interface to configure key pairs for signing and verifying tokens.

### About this task

Manage Keystore assignment and Keystore history for tokens

### Procedure

1. In the Cloud Manager, click .
2. In the Settings navigation list, click Signed Token.  
The Signed Token page displays the list of supported tokens for signing.
3. Review the current token settings:
  - Click Payload to display the payload code for the set of tokens.
  - Click History log next to a token to view its previous keystore assignments (this history never includes the current keystore assignment). The most recent assignment displays at the top of the list. You can remove any keystore assignment from the history by clicking .
4. To add or change a keystore assignment for a token, complete the following steps:
  - a. On the Signed Token page, click Edit.
  - b. On the Edit signed token page, select a keystore for the token you want to modify.
  - c. Click Save to save your changes and update the keystore history.  
Changes take effect immediately.

The previously selected keystore is added to the history for the token that you just updated. The history never includes the current keystore assignment.

---

## Using the Toolkit CLI to configure public/private key pairs

Use the Toolkit CLI to configure key pairs for signing and verifying tokens.

### About this task

Use the following properties to the keystore URLs that contain the public/private key pair for each type of token. For each setting, the list of keystores is an array of strings, with each string representing the complete URL of a keystore. The first keystore in the array (`keystore_url_0`) is the one that is currently in use.

- `access_token`:  

```
access_token_keystore_urls: [keystore_url_0, keystore_url_n]
```
- `id_token`:  

```
id_token_keystore_urls: [keystore_url_0, keystore_url_n]
```
- `temporary_token`:  

```
temporary_token_keystore_urls: [keystore_url_0, keystore_url_n]
```

The keystore URLs can be lengthy. You can avoid errors by writing the settings in a file and uploading that file. Format the settings as a .yaml file (you can choose the file name) and upload the file using the toolkit CLI. The following example configures a different keystore for each type of token:

```
access_token_keystore_urls:
- >-
 https://mgmt_endpoint_url/api/orgs/1cdb71a1-b53c-4b66-b162-1b0f45ca8a54/keystores/de5298dd-a00b-45a0-9d08-96c5d93e1273
id_token_keystore_urls:
- >-
 https://mgmt_endpoint_url/api/orgs/1cdb71a1-b53c-4b66-b162-1b0f45ca8a54/keystores/85a21745-8ac3-499d-929a-2d017bade815
temporary_token_keystore_urls:
- >-
 https://mgmt_endpoint_url/api/orgs/1cdb71a1-b53c-4b66-b162-1b0f45ca8a54/keystores/63d6462f-c93d-439b-b422-0e604381e97a
```

## Procedure

1. Create a YAML file with the keystore settings for each type of token as shown in the example.  
For information on using input files with the toolkit CLI, see [Reading input from the command line](#).
2. Run the following command to log in to a management server as a member of the cloud administration organization:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm admin/identity_provider
```

Cloud settings are stored on a management server and you must log in to the server before you can access the settings with the CLI.

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`. For more information on logging in, see [Logging in to a management server](#).

3. Upload the YAML file with the new settings by running the following command:

```
apic cloud-settings:update --server mgmt_endpoint_url my_token_keystore_settings.yaml
```

4. Run the following command to log out of the management server:

```
apic logout --server mgmt_endpoint_url
```

---

## Administering provider organizations

Manage the provider organizations who have accounts to publish APIs in your API Connect cloud.

### About this task

---

Perform these tasks to manage your provider organizations:

- [Creating a provider organization](#)  
For developers to publish APIs, they must be a member of a provider organization. The first step is to create a provider organization and specify an organization owner. Then members can be added to the provider organization and can start publishing APIs.
- [Editing a provider organization](#)  
You can edit the title of a provider organization, and also transfer ownership to a new organization owner.
- [Deleting a provider organization](#)  
You can delete a provider organization. All Catalogs and Spaces are also removed.
- [Viewing current provider organizations](#)  
View the list of current provider organizations for your cloud, including organization title, owner's name and email address, and status. From the list screen, you can add new provider organizations and owners, edit and delete provider organizations, send messages, and change owners.
- [Sending messages to provider organization owners](#)  
You can send an email message to the owner of a provider organization. You can also style the email to match your business theme.

### Related tasks

---

- [Creating a provider organization](#)
- [Editing a provider organization](#)
- [Deleting a provider organization](#)
- [Viewing current provider organizations](#)

---

## Creating a provider organization

For developers to publish APIs, they must be a member of a provider organization. The first step is to create a provider organization and specify an organization owner. Then members can be added to the provider organization and can start publishing APIs.

## Before you begin

---

This task can be completed by users who are assigned one of the following roles:

- Cloud Owner
- Cloud Administrator
- Organization Manager
- Custom role with the `Provider-org:Manage` permission

You must also complete the following tasks before beginning:

- [Configuring an email server for notifications](#)
- [Setting up notifications](#)

## About this task

---

Note that an email server must be configured and an active email server must be selected before a provider organization account can be created.


There are two options for creating a provider organization account, as follows:

- Option 1: Create a provider organization and send an email invitation to a new user to register and become the organization owner. Choose the Add > Invite organization owner option.
- Option 2: Create a provider organization and specify the user who will be the organization owner. Choose the Add > Create organization option.

## Procedure

---


- **Option 1:** To create a provider organization and invite a new user to register as the owner, complete the following steps:

1. In the Cloud Manager, click  Provider Organizations.
2. Click Add > Invite organization owner
3. Enter a valid email address and click Invite. An email containing an activation link is sent to the email address inviting the person to register. The link opens the registration form. Alternately, you can copy and paste the activation link into an email; click Get Activation Link and click Copy in the pop up window. Note that the email server requires the Authenticate User and Authenticate Password values in order to send invitations. See [Configuring an email server for notifications](#).

The new provider organization is placed in Pending status, and is changed to Enabled status after the owner completes the registration form to activate their account. The owner can then log in to the API Manager user interface with the API Connect user name that they specified during account activation.

Note: If you use this option with a Local User Registry, and the user already exists in the registry, the user must re-activate their account by using the Sign In option, **not** by completing the registration form and using the Sign Up option; attempting to re-register will fail. This stipulation includes users that were previously provider organization owners but whose organization was deleted.

- **Option 2:** To create a provider organization and specify the user who will be the organization owner, complete the following steps:

1. In the Cloud Manager, click  Provider Organizations.
2. Click Add > Create organization.
3. In the Title field, enter a title for the provider organization. If desired, the title can be edited after the provider organization is created.
4. The internal Name is automatically generated for you by the system. The Name field contains the string that is included in the organization segment of the URL for API calls. For more information, see [Calling an API](#).

Tip: The Name field is auto-generated to ensure its validity in the URL.

The value in the Name field is also used to identify the provider organization in developer toolkit CLI commands. To view the CLI commands to manage provider organizations, see [apic.orgs](#).

5. Select the user registry for the provider organization owner.

The remaining procedure varies according to the type of the selected user registry, as follows:

- Local User Registry
  - Select whether the user is an Existing user or a New User.
    - For an existing user, complete the following steps:
      - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account.
      - Click Create. The provider organization is created and immediately enabled. The specified owner can log in to the API Manager user interface. If the new provider organization is the user's only provider organization, the API Manager user interface opens in that provider organization; if the user is a member of more than one provider organization, they can then select the new provider organization from the Organization list.
    - For a new user, complete the following steps:
      - Enter a unique user name for the new user.
      - Supply an email address, name details, and a password.
      - Click Create. The user account is created, and the provider organization is created and immediately enabled. The specified owner can log in to the API Manager user interface with the specified user name and password; the user interface open in the new provider organization.
- LDAP
  - Enter the name of a user that exists in the selected user registry.
  - Click Create. The provider organization is created and immediately enabled. The specified owner can log in to the API Manager user interface with their LDAP user name. If the new provider organization is the user's only provider organization, the API Manager user interface opens in that provider organization; if the user is a member of more than one provider organization, they can then select the new provider organization from the Organization list.
- Authentication URL and OIDC
  - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account. If the required organization owner has not previously been invited and activated their account, you must use the invitation mechanism described in [Option 1](#).

- Click Create. The provider organization is created and immediately enabled. The specified owner can log in to the API Manager user interface. If the new provider organization is the user's only provider organization, the API Manager user interface opens in that provider organization; if the user is a member of more than one provider organization, they can then select the new provider organization from the Organization list.

## What to do next

---

For details of how an invited organization owner activates their account see [Activating your API Manager user account](#). For details of how to log in to the API Manager user interface, see [Accessing the API Manager user interface](#).

## Editing a provider organization

---

You can edit the title of a provider organization, and also transfer ownership to a new organization owner.

### Before you begin

---

This task can be completed by users who are assigned the following roles:

- Cloud Owner
- Cloud Administrator
- Organization Manager
- Custom role with the `Provider-org:Manage` permission

### About this task


---

If you transfer ownership, the new organization owner can be an existing user, or you can create a new user. When the ownership changes, the new owner is assigned administration privileges for the organization, and the previous owner has their administration privileges removed. Optionally, you can also have the ownership of all Catalogs and Spaces in the organization transferred to the new organization owner. If required, you can restore the privileges to the previous owner as described in [Administering members and roles](#).

### Procedure

---

To update a provider organization, complete the following tasks:

1. In the Cloud Manager, click  Provider Organizations.
2. Click the name of the provider organization that you want to update.
3. To change the title, enter a new title in the Title field, then click Save.
4. To transfer ownership, select the user registry for the provider organization owner you want to transfer ownership to.  
The remaining procedure varies according to the type of the selected user registry, as follows:
  - Local User Registry
    - Select whether the user is an Existing user or a New User.
      - For an existing user, complete the following steps:
        - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account.
        - To also have the ownership of all Catalogs and Spaces in the organization transferred to the new organization owner, select Also transfer ownership of owned Catalogs and Spaces.
        - Click Save.
      - For a new user, complete the following steps:
        - Enter a unique user name for the new user.
        - Supply an email address, name details, and a password.
        - To also have the ownership of all Catalogs and Spaces in the organization transferred to the new organization owner, select Also transfer ownership of owned Catalogs and Spaces.
        - Click Save.
  - LDAP
    - Enter the name of a user that exists in the selected user registry.
    - To also have the ownership of all Catalogs and Spaces in the organization transferred to the new organization owner, select Also transfer ownership of owned Catalogs and Spaces.
    - Click Save.
  - Authentication URL and OIDC
    - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account. If the required organization owner has not previously been invited and activated their account, you must ensure they have been invited to the organization first.
    - To also have the ownership of all Catalogs and Spaces in the organization transferred to the new organization owner, select Also transfer ownership of owned Catalogs and Spaces.
    - Click Save.

## Deleting a provider organization

---

You can delete a provider organization. All Catalogs and Spaces are also removed.

### Before you begin

---

This task can be completed by users who are assigned the following roles:

- Cloud Owner
- Cloud Administrator
- Organization Manager
- Custom role with the `Provider-org:Manage` permission

## About this task

---


The user account for the owner of the deleted provider organization remains in the associated API Connect user registry. The user can still log in to the API Manager user interface; if they are a member of one or more other provider organizations then those organizations are accessible, otherwise only information links are available, with no access to view or manage any resources.

The user can be added again as a provider organization owner at a later time; see [Creating a provider organization](#).

## Procedure

---

To delete a provider organization, complete the following tasks:

1. In the Cloud Manager, click  Provider Organizations.
2. Navigate to the provider organization that you want to delete in the list and select Delete from the Actions menu.
3. Confirm that you want to delete the provider organization.  
The provider organization is deleted and removed from the Provider Organizations list.

## Results

---

The provider organization, and its Catalogs and Spaces, are deleted.

## Viewing current provider organizations

---

View the list of current provider organizations for your cloud, including organization title, owner's name and email address, and status. From the list screen, you can add new provider organizations and owners, edit and delete provider organizations, send messages, and change owners.

## Before you begin

---

You must also complete the following tasks before beginning:

- [Configuring an email server for notifications](#)
- [Setting up notifications](#)
- [Creating a provider organization](#)

This task can be completed by users who are assigned one of the following roles:

- All roles
- Custom role with the `Provider-org:Manage` or the `Provider-org:View` permissions


## About this task

---

All roles can view the list of provider organizations. Use the provider organization list screen to manage your provider organizations.

## Procedure

---

1. In the Cloud Manager, click  Provider Organizations.
2. View the list of provider organizations and their status. The statuses are as follows:
  - **Pending:** A provider organization is in `Pending` status until the invited owner successfully completes the registration form.
  - **Active:** A provider organization is in `Active` status if the owner is an existing user with an account in a user registry.
3. The Actions menu performs different functions depending on the organization's status. Use the Actions menu to edit or delete an Active organization. For Pending organizations, use the Actions menu to delete the organization and to resend the invitation to the owner
  - For an Enabled organization, you can Edit or Delete the organization.
  - For a Pending organization, you can Resend the Invitation, or Delete the request.
4. Add new provider organizations using the Add menu. See [Creating a provider organization](#)

## Related tasks

---

- [Creating a provider organization](#)
- [Editing a provider organization](#)
- [Deleting a provider organization](#)

## Sending messages to provider organization owners

---

You can send an email message to the owner of a provider organization. You can also style the email to match your business theme.

## Before you begin

This task can be completed by users who are assigned the following roles:

- Cloud Owner
- Cloud Administrator
- Organization Manager
- Custom role with the `Provider-org:Manage` permission

Note: You can also send email messages by using the developer toolkit CLI, or by using the API Connect REST APIs. See [Cloud administration commands](#), or [API Connect REST APIs](#), for more information.

## Procedure

To send a message, complete the following steps:



1. In the Cloud Manager, click  Provider Organizations.
2. Click the options icon  alongside the provider organization that you want to work with, then click Message owner. The Send message window opens, and the email address of the provider organization owner is displayed as the recipient.
3. Enter the subject of the message in the Subject field.
4. Select the Content type that you want to use for the entering the content of the message, from HTML, PlainText, or Both. The default content type is PlainText. An edit window for the selected content type is displayed, or both edit windows are displayed if Both is selected. For HTML content, only the tags and their attributes that are shown in the following table are allowed.

Table 1. List of allowed HTML tags and their attributes

HTML tag	Attribute
<a>	"class", "href", "hreflang", "style"
<em>	"class", "style"
<strong>	"class", "style"
<cite>	"class", "style"
<blockquote>	"class", "cite", "style"
<code>	"class", "style"
<ul>	"class", "type", "style"
<ol>	"class", "start", "type", "style"
<li>	"class", "style"
<dl>	"class", "style"
<dt>	"class", "style"
<dd>	"class", "style"
<h1>	"class", "id", "style"
<h2>	"class", "id", "style"
<h3>	"class", "id", "style"
<h4>	"class", "id", "style"
<h5>	"class", "id", "style"
<h6>	"class", "id", "style"
<p>	"class", "style"
<div>	"class", "style"
 	"class", "style"
<span>	"class", "style"
<img>	"class", "src", "alt", "data-entity-type", "data-entity-uuid", "data-align", "data-caption", "width", "height", "style"
<table>	"class", "id", "style"
<tr>	"class", "id", "style"
<td>	"class", "id", "style"

If an HTML tag that is not allowed is used in a notification, the tag and its contents are displayed in the email as plain text.

Images can be used by adding a `<img`

`src="https://path/to/image.png"/>` tag in the template. The `src` attribute for the image must be a fully qualified web URL, and must be externally accessible so that the email recipients can access the image. It's not possible to reference local images, they must be fully qualified URLs. It's also not possible to embed or attach images or other files in the emails.

5. Enter the content of the message.
6. Click Send.

## Administering members and roles

Cloud Administrators can add members and assign them roles to enable them to work in Cloud Manager. The Cloud and Topology Administrators can also create roles and role defaults. You can also delete users to prevent them from accessing Cloud Manager.

## About this task

To manage roles, add members, and assign roles to member, perform the following tasks:

- [Creating roles in the admin organization](#)  
Cloud Manager ships with a set of pre-configured roles. As the Cloud Manager administrator, you can create custom roles, add permissions, and assign the roles to members of the Admin organization.
- [Editing roles in the admin organization](#)  
As the Cloud Manager administrator, you can edit the roles in the Admin Organization. Both pre-configured roles and custom roles may be edited, except for the

Owner, Viewer, and Member roles. The Owner, Viewer, and Member roles may not be edited or deleted.

- [Configuring LDAP group mappings on Cloud Manager user roles](#)

As a Cloud Manager administrator, you can configure LDAP group mapping on API Connect roles in the Admin organization by using the developer toolkit CLI.

- [Deleting roles in the admin organization](#)

Roles can be deleted (except for the Owner, Viewer, and Member roles) from the Admin organization. If there are members assigned to a role that has been deleted, they will be assigned the Viewer role.

- [Viewing members and roles](#)

Use the Members page to view the current members of the Admin organization and manage the role assignments.

- [Adding members to the admin organization](#)

As the Cloud Manager administrator, you can add and remove members of the Admin organization and assign roles to them. Once removed, a user can no longer access the Cloud Manager.

- [Assigning roles to members](#)

As the Cloud Manager administrator, you manage the roles that are assigned to members. A member's role determines the tasks they are authorized to perform in Cloud Manager.

- [Deleting a member](#)

As the Cloud Manager administrator, you can delete members of the API Connect cloud administration organization. Once deleted, the user and associated roles are removed, however the user's account still remains in API Connect.

- [Role Defaults overview](#)

Role Defaults are role templates that determine the default roles created in new Provider and Consumer organizations. Cloud Manager ships with a set of pre-configured Role Defaults. As the Cloud Manager administrator, you can edit the pre-configured Role Defaults.

---

## Creating roles in the admin organization

Cloud Manager ships with a set of pre-configured roles. As the Cloud Manager administrator, you can create custom roles, add permissions, and assign the roles to members of the Admin organization.

---

### About this task

If you are assigned a role that has the `Settings: manage` permission, you can create custom roles for the Admin organization. The Admin organization roles apply only to Cloud Manager users.


One of the following roles is required to add custom roles:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: manage` permissions

---

### Procedure

Perform the following steps to add custom roles to the Admin Organization:

1. In the Cloud Manager, click  Settings.
2. Select Roles.
3. Click Add to add a new role.
4. Enter the Title for the role. The Name will be auto-generated.
5. Use the check boxes to select the permissions for the role. For a description of the permissions and the actions they enable, see [User roles and permissions in the Cloud Manager UI](#).
6. Save the custom role.

---

### Results

The custom role appears in the list of roles and can be assigned to members.

---

### What to do next

Assign the role to a member in the Members list. See [Assigning roles to members](#)

---

### Related tasks

- [Editing roles in the admin organization](#)

---

## Editing roles in the admin organization

As the Cloud Manager administrator, you can edit the roles in the Admin Organization. Both pre-configured roles and custom roles may be edited, except for the Owner, Viewer, and Member roles. The Owner, Viewer, and Member roles may not be edited or deleted.

---

### About this task

You can customize the per-configured roles (except for the Owner, Viewer, and Member roles) to reflect your business needs.



If you are assigned a role that has the `Settings: manage` permission, you can edit the roles for the Admin organization. The Admin organization roles apply only to Cloud Manager users.

One of the following roles is required to edit roles:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: manage permissions`


Following are the edits you can perform for a role:

- Change the title
- Add or delete permissions

## Procedure

---

Perform the following steps to edit roles in the Admin organization:

1. In the Cloud Manager, click  Settings.
2. Select Roles.
3. Locate the Role you want to edit, and select Edit from the overflow menu.
4. Enter the desired Title for the role. The Name will be auto-generated.
5. Use the check boxes to select the permissions for the role. For a description of the permissions and the actions they enable, see [User roles and permissions in the Cloud Manager UI](#).
6. Save the updated role.

## Results

---

The Role will be updated with a new name and/or permissions. An updated role will affect the members who are assigned the role. If you change the name of a role, you will have to assign it to the members. If you change the permissions for a role, the way that members use the product may change.

## Configuring LDAP group mappings on Cloud Manager user roles

---

As a Cloud Manager administrator, you can configure LDAP group mapping on API Connect roles in the Admin organization by using the developer toolkit CLI.

### Before you begin

---

You must have an LDAP user registry resource in the Cloud Manager that has the `external_group_mapping_enabled` configuration set to `true`. See [Using the CLI to configure a shared LDAP user registry](#), for information.

One of the following roles is required to edit roles:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: manage permissions`

Note: The Admin organization roles apply only to Cloud Manager users. If you want to apply LDAP group mapping to API Manager users, see [Configuring LDAP group mapping on API Manager user roles](#). LDAP group mapping cannot be applied to Developer Portal users.

### About this task

---

You can map external LDAP groups to the API Connect preconfigured user roles (except for the Owner and Member roles), as well as to any custom user roles, to reflect your business needs.

Notes:

- After LDAP group mapping is enabled on a role, user onboarding always honors the group mappings.
- Once on-boarded, user membership in API Connect is valid throughout the login period (`access_token_ttl`), irrespective of any changes in the external LDAP registry. Membership is updated only on the next login, when the LDAP information is fetched and refreshed.
- One or more API Connect roles can be mapped to one or more LDAP groups, and one or more LDAP groups can be mapped to a role.
- When multiple LDAP groups are mapped to a single role, it means that a user from any one of the LDAP groups can log in to API Connect.
- If a user is removed from the external LDAP user registry, then to ensure quick removal from API Connect you must also delete the user membership in API Connect.

You can configure LDAP group mappings using one of the following methods:

- [Using the UI to configure LDAP group mappings](#)
- [Using the CLI to configure LDAP group mappings](#)

## Using the UI to configure LDAP group mappings

---

Use the Cloud Manager UI to configure LDAP group mappings.

### Procedure

1. Login to the Cloud Manager UI as a user with the appropriate permissions.
2. Click Settings > Roles > Add a new Role or edit existing role .
3. Select the following settings:
  - Enable external group mapping - select to enable external group mapping for this role.
  - LDAP user Registry - LDAP user registry list which has external group mapping enabled. User must select one of the LDAP user registry from this list.
  - LDAP group names - add one or more LDAP group names that you want to map to the user role.
  - User group prefix - add the prefix for the LDAP user group.
  - User group suffix - add the suffix for the LDAP user group.
4. Save your changes.

## Results

The role is updated with the LDAP group mapping information. Users can now log on to the Cloud Manager, and automatically be assigned the correct access permissions for their role.

## Using the CLI to configure LDAP group mappings

Use the toolkit CLI to configure LDAP group mappings.

### About this task

For detailed information about how to use the CLI, see [Installing the toolkit](#), and [Overview of the command-line tool](#).

### Procedure

Perform the following steps to map external LDAP groups to Cloud Manager user roles.

1. Log in to the management server CLI.  
Before you can update the role configuration, you must log in to your management server from the developer toolkit CLI as a member of the cloud administration organization. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm admin/identity_provider
```

where `mgmt_endpoint_url` is the platform API endpoint URL.

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`.

For full details of the login command, see [Logging in to a management server](#).

2. Run the following command to get the URL of the LDAP user registry resource in the Cloud Manager that you want to map the user roles to:

```
apic user-registries:get ldap_user_registry --org admin --server mgmt_endpoint_url --output -
```

where:

- `ldap_user_registry` is the name or ID of your LDAP user registry resource.
- `--org admin` means that the registry details are retrieved from the admin organization.
- `mgmt_endpoint_url` is the platform API endpoint URL.

This command outputs the configuration details of your LDAP user registry, and the `url` is shown at the end of the list, for example:

```
type: user_registry
api_version: 2.0.0
id: 35e75bad-1d89-4a65-a70f-xxxxxx
name: ldap
title: LDAP
integration_url: >-
 https://server.com/api/cloud/integrations/user-registry/147b5fb1-e88e-41e3-90e9-xxxxxx
registry_type: ldap
user_managed: false
user_registry_managed: false
external_group_mapping_enabled: true
...
url: >-
 https://server.com/api/user-registries/3d58ce7e-16a8-493b-9684-xxxxxx/35e75bad-1d89-4a65-a70f-xxxxxx
```

3. Create a role yaml file that contains the following LDAP group mapping configuration properties:

```
external_group_mapping:
 user_registry_url: https://server.com/api/user-registries/3d58ce7e-16a8-493b-9684-xxxxxx/35e75bad-1d89-4a65-a70f-xxxxxx
 ldap_groups:
 - 'cn=apic-administrators,ou=ibmgroups,o=ibm.com'
 - 'cn=apic-developers,ou=ibmgroups,o=ibm.com'
 user_group_filter_prefix: (&(uniquemember=
 user_group_filter_suffix:) (objectClass=groupOfUniqueNames))
```

Where:

- `user_registry_url` is the URL of your LDAP user registry resource from Step [#task\\_cmc\\_role\\_mapping\\_ldap\\_url](#).
- `ldap_groups` is a list of the LDAP group names that you want to map to the user role.
- `user_group_filter_prefix` is the prefix for the LDAP user group.
- `user_group_filter_suffix` is the suffix for the LDAP user group.

4. Run the following command to update the user role with the `external_group_mapping` configuration properties:

```
apic roles:update role_name --scope org --org admin? --server mgmt_endpoint_url mapping_properties_file
```

Where:

- `role_name` is the name of the user role that you want to add the LDAP group mapping to.
- `--scope` is the organization scope that you want the update to apply to. Valid values are: `catalog|consumer-org|org|space`.
- `--org admin?`
- `mgmt_endpoint_url` is the platform API endpoint URL.
- `mapping_properties_file` is the name of your mapping properties file from Step [#task\\_cmc\\_role\\_mapping\\_role\\_file](#), for example `role_mapping_file.yaml`.

If you prefer to enter the configuration properties interactively on the command line, you can substitute the `mapping_properties_file` for a terminating hyphen character `-`, and enter the information manually, followed by pressing **CTRL** **D** to terminate the input.

If you want to create a custom role that includes LDAP group mapping, you can include the `external_group_mapping` configuration section in the `role_file`, and then create the new role by using the `apic roles:create` command.

For more information about the `apic roles` commands, see [apic roles](#) in the CLI reference section.

## Results

The role is updated with the LDAP group mapping information. Users can now log on to the Cloud Manager, and automatically be assigned the correct access permissions for their role.

---

## Deleting roles in the admin organization

Roles can be deleted (except for the Owner, Viewer, and Member roles) from the Admin organization. If there are members assigned to a role that has been deleted, they will be assigned the Viewer role.

## About this task

All custom roles may be deleted. Pre-configured roles can also be deleted, except for the Owner, Viewer, and Member roles. Owner, viewer, and Member role cannot be deleted or edited. Members who are assigned a deleted role are given the most limited role, Viewer, with View-only privileges. These members will need to be assigned a new role if they require any other privileges.


If you are assigned a role that has the `Settings: manage` permission, you can delete roles for the Admin organization. The Admin organization roles apply only to Cloud Manager users.

One of the following roles is required to delete custom roles:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: manage permissions`

## Procedure

Perform the following steps to delete roles in the Admin organization:

1. In the Cloud Manager, click  Settings.
2. Select Roles.
3. Locate the Role you want to delete, and select Delete from the overflow menu.
4. Confirm that you want to proceed with deleting the role.

## Results

The Role will be deleted. Members who had been assigned the role are given the Viewer role.


---

## Viewing members and roles

Use the Members page to view the current members of the Admin organization and manage the role assignments.

## Procedure

To view the members of the Admin organization and their roles in Cloud Manager, follow these steps:

1. In the Cloud Manager, click  Members.

- From the Members list, you can scroll through the list of users to see their current roles and their status.
    - Pending** - The invitation with the activation link has been sent to the user, but they have not yet completed the registration form.
    - Enabled** - The user has completed the registration form and is now able to login and access the features associated with their role.
- Note: If your role has the proper permissions, you can assign roles, add, and delete members from this list.

## What to do next

---

- To add members, see [Adding members to the admin organization](#)
- To change the roles assigned to members see [Assigning roles to members](#)
- To delete a member, see [Deleting a member](#)

## Adding members to the admin organization

---

As the Cloud Manager administrator, you can add and remove members of the Admin organization and assign roles to them. Once removed, a user can no longer access the Cloud Manager.

### About this task

---

Membership in the Admin organization is required to use Cloud Manager to view and manage the topology of the API Connect cloud, to set up and manage provider organizations, and to manage other features of the cloud. There are two ways to add a member:


- Invite member - send an email invitation to a new user to register as a member of the Admin organization.
- Add member - specify the user and add them as a member of the Admin organization immediately.

One of the following roles is required to add members:

- Administrator
- Owner


### Procedure

---

- To send an email invitation to a new user to register as a member of the Admin organization, complete the following steps:
  - In the Cloud Manager, click  Members.  
The list of current members for the Admin organization is displayed. You can view the owner of the Admin Organization by expanding View Owners.
  - Click Add > Invite member.
  - Enter a valid email address.
  - Select the roles that you want to assign to the user.
  - Click Invite. An email containing an activation link is sent to the email address inviting the person to register. The link opens the registration form. Alternately, you can copy and paste the activation link into an email; click Get Activation Link and click Copy in the pop up window.  
Note that the email server requires the Authenticate User and Authenticate Password values in order to send invitations. See [Configuring an email server for notifications](#).

The user is placed in Pending status, and is changed to Enabled status after they complete the registration form to activate their account. They can then log in to the Cloud Manager user interface with the API Connect user name that they specified during account activation.

Note: If you use this option with a Local User Registry, and the user already exists in the registry, the user must re-activate their account by using the Sign In option, **not** by completing the registration form and using the Sign Up option; attempting to re-register will fail. This stipulation includes users that were previously members but were deleted from the admin organization.

- To specify the user and add them as a member of the Admin organization immediately, complete the following steps:
  - In the Cloud Manager, click  Members.  
The list of current members for the Admin organization is displayed. You can view the owner of the Admin Organization by expanding View Owners.
  - Click Add > Add member.
  - Select the required user registry  
The remaining procedure varies according to the type of the selected user registry, as follows:
    - Local User Registry
      - Select whether the user is an Existing user or a New User.
        - For an existing user, complete the following steps:
          - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account.
          - Select the roles that you want to assign to the user, then click Add. The user is added and their membership is immediately enabled. The specified user can log in to the Cloud Manager user interface.
        - For a new user, complete the following steps:
          - Enter a unique user name for the new user.
          - Supply an email address, name details, and a password.
          - Select the roles that you want to assign to the user, then click Add. The user account is created, and the user membership is immediately enabled. The specified user can log in to the Cloud Manager user interface.
    - LDAP
      - Enter the name of a user that exists in the selected user registry.
      - Select the roles that you want to assign to the user, then click Add. The user is added, and the user membership is immediately enabled. The specified user can log in to the Cloud Manager user interface with their LDAP user name.
    - Authentication URL and OIDC
      - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account.
      - Select the roles that you want to assign to the user, then click Add. The user is added, and the user membership is immediately enabled. The specified user can log in to the Cloud Manager user interface.

## What to do next

---

The new member can log in and work in Cloud Manager. The member's authorization is defined by the roles assigned to them. For instructions on assigning roles, see [Assigning roles to members](#). The list of roles and permissions can be viewed at [User roles and permissions in the Cloud Manager UI](#).

---

## Assigning roles to members

As the Cloud Manager administrator, you manage the roles that are assigned to members. A member's role determines the tasks they are authorized to perform in Cloud Manager.

---

### About this task

Members in the Admin organization use Cloud Manager to view and manage the topology of the Cloud, to set up and manage provider organizations, and to work with other features of the API Connect Cloud. The member's role determines what tasks they are authorized to perform. When assigning roles, you need to be aware that all members always have a basic set of permissions, which are contained in the **Member** role. The Member role is automatically granted to all members and cannot be removed. For this reason, you do not explicitly assign the Member role. You can grant additional permissions by selecting one or more roles. If additional roles are not assigned, then the user will have only the basic Member role.


One of the following roles is required to assign roles to members:

- Administrator
- Owner

---

### Procedure

Perform the following steps to assign one or more roles to a member:

1. In the Cloud Manager, click  Members.
2. The list of current members for the Admin organization is displayed.
3. To change the roles assigned to a member, select the roles by marking the check boxes next to their name. The roles will be updated instantly. The list of roles and permissions can be viewed at [User roles and permissions in the Cloud Manager UI](#)

---

### Results

The member will be assigned the permissions for the role(s).

---

### What to do next

The new member can work in Cloud Manager. The member's authorization is defined by the roles assigned to them.

---

## Deleting a member

As the Cloud Manager administrator, you can delete members of the API Connect cloud administration organization. Once deleted, the user and associated roles are removed, however the user's account still remains in API Connect.

---

### About this task

One of the following roles is required to delete a member:


- Administrator
- Owner

Note: The user account of the deleted user remains in the associated API Connect user registry. The user can still log in to the Cloud Manager user interface but only information links are available; there is no access to view or manage any resources. The user can be added again as a member at a later time; see [Adding members to the admin organization](#).

---

### Procedure

To delete a member of the cloud administration organization, complete the following steps:

1. In the Cloud Manager, click  Members.
2. Choose Delete from the overflow menu that is adjacent to the member you want to delete.
3. Confirm the deletion.

---

### Results

The member is deleted and removed from the Members list in Cloud Manager.

---

## Role Defaults overview

Role Defaults are role templates that determine the default roles created in new Provider and Consumer organizations. Cloud Manager ships with a set of pre-configured Role Defaults. As the Cloud Manager administrator, you can edit the pre-configured Role Defaults.

## Role Defaults concepts

---

Role Defaults determine the base roles applied to either Provider or Consumer Organizations. The Role Defaults for the Admin organization are not configurable. Role Defaults help to standardize base roles for all organizations. Roles and permissions may be added or modified in the Role Defaults to customize the base roles as needed. Role Defaults are a "point-in-time" configuration, which effects only new organizations. You can add, delete, or modify roles (except for the Owner, Member, and Viewer roles) in the organization to customize the permissions for each organization.

Rules for editing Role Defaults:

- Edits to Role Defaults are not applied retroactively to existing organizations. Edits apply only to new organizations created after the changes are made.
- Role Defaults apply to all new organizations; roles can then be edited at the organization, catalog, and space level.
- Edits made to roles in an organization do not affect the Role Defaults. Role Defaults are the "parent" and organization roles are the "child". The child inherits from the parent, but changes do not filter up from the organization to the Role Default.
- [Managing role defaults for provider organizations](#)  
Role Defaults are role templates that determine the default roles created in new Provider and Consumer organizations. Cloud Manager ships with a set of pre-configured Role Defaults. As the Cloud Manager administrator, you can edit the pre-configured Role Defaults.
- [Managing role defaults for consumer organizations](#)  
Role Defaults are role templates that determine the default roles created in new Provider and Consumer organizations. Cloud Manager ships with a set of pre-configured Role Defaults. As the Cloud Manager administrator, you can edit the pre-configured Role Defaults.

## Related tasks

---

- [Managing role defaults for provider organizations](#)
- [Managing role defaults for consumer organizations](#)

## Related information

---

- [Creating custom roles](#)

## Managing role defaults for provider organizations

---

Role Defaults are role templates that determine the default roles created in new Provider and Consumer organizations. Cloud Manager ships with a set of pre-configured Role Defaults. As the Cloud Manager administrator, you can edit the pre-configured Role Defaults.

## About this task

---


If you are assigned a role that has permissions to manage Cloud Settings, you can create manage the Role Defaults for the provider organizations. One of the following roles is required to add roles to the role defaults:

- Administrator
- Topology Administrator
- Owner

Perform the following steps to add or edit roles to the Roles Defaults for provider organizations:

## Procedure

---

1. In the Cloud Manager, click  Settings.
2. Select Role Defaults.
3. Click Add to add a new role.
  - a. To edit a role, select Edit from the overflow menu for the role.
  - b. To delete a role from the Role Default, select Delete from overflow menu and confirm the deletion. Owner, Member, and Viewer roles cannot be deleted.  
Note: Role deletions apply to new members added after the deletion. If a current member is assigned the role, they will continue to have the permissions connected with the role. You need to manage the deletions for each member in the Members screen.
4. Enter the Title for the role. The Name will be auto-generated.
5. Use the check boxes to select the permissions for the role. For a description of the permissions and the actions they enable, see [User roles and permissions in the Cloud Manager UI](#).
6. Save the role.

## Results

---

The role appears in the list of roles for the provider organization Role Defaults. It will be included in the roles that can be assigned to members for all new provider organizations. The roles can be edited for individual organizations.

## Related concepts

---

- [Role Defaults overview](#)

## Related tasks

---

- [Managing role defaults for consumer organizations](#)
- [Assigning roles to members](#)

## Managing role defaults for consumer organizations

---

Role Defaults are role templates that determine the default roles created in new Provider and Consumer organizations. Cloud Manager ships with a set of pre-configured Role Defaults. As the Cloud Manager administrator, you can edit the pre-configured Role Defaults.

### About this task

---


If you are assigned a role that has permissions to manage Cloud Settings, you can create and manage the Role Defaults for consumer organizations. One of the following roles is required to add roles to the role defaults:

- Administrator
- Topology Administrator
- Owner

Perform the following steps to add or edit roles to the roles defaults for consumer organization:

### Procedure

---

1. In the Cloud Manager, click  Settings.
2. Select Role Defaults.
3. Click Add to add a new role.
  - a. To edit a role, select Edit from the overflow menu for the role.
  - b. To delete a role from the Role Default, select Delete from overflow menu and confirm the deletion. Owner, Member, and Viewer roles cannot be deleted.  
Note: Role deletions apply to new members added after the deletion. If a current member is assigned the role, they will continue to have the permissions connected with the role. You need to manage the deletions for each member in the Members screen.
4. Enter the Title for the role. The Name will be auto-generated.
5. Use the check boxes to select the permissions for the role. For a description of the permissions and the actions they enable, see [User roles and permissions in the Cloud Manager UI](#).
6. Save the role.

### Results

---

The role appears in the list of roles for the consumer organization Role Defaults. It will be included in the roles that can be assigned to members for new consumer organizations. The roles can be edited for catalogs, but permissions inherited from the Role Default cannot be deleted. Permissions can be added at the catalog level, but not deleted.

### Related concepts

---

- [Role Defaults overview](#)

## Related tasks

---

- [Managing role defaults for provider organizations](#)
- [Assigning roles to members](#)

## Monitoring the cloud

---

API Connect generates events to monitor the status of your cloud.

### About this task

---

You can monitor your API Connect cloud and configure destination targets for your analytics data by using the following tasks:

- [Accessing and configuring analytics](#)  
Monitor the usage and performance of API calls by using the analytics feature to view, filter, sort, and aggregate your API event data.
- [Configuring audit logging to monitor user operations](#)  
Configure auditing in IBM API Connect to monitor user operations such as who published which product and when.
- [Reviewing a gateway's processing status](#)  
Review the processing status for the Gateway services that are enabled in your API Connect Catalogs and Spaces.

## Accessing and configuring analytics

---

Monitor the usage and performance of API calls by using the analytics feature to view, filter, sort, and aggregate your API event data.

Use the API Connect Analytics service to monitor API usage and performance. API analytics provide data to help you manage service levels, set quotas, establish controls, set up security policies, manage communities, and analyze trends. The data for Analytics is collated from API events that are logged when API operations are invoked on your gateways. The captured event data can be viewed and exported from the Cloud Manager UI, API Manager UI, toolkit CLI or REST API. The API event data can also be offloaded to third-party systems as it is generated.

## API event records

An API event is logged each time an API operation is invoked on the Gateway server. The API event record contains information about the API call. When an analytics service is associated with a gateway, all API invocations on that gateway are captured as API event records and sent to the analytics service. The content of the record depends on the logging policy that is set for the operation. For more information about the fields that are displayed in an API event record, see [API event record fields](#).

## Analytics configuration options

API Connect offers several configuration options to customize data collection, storage, and retention. For example, you can define filters that refine data to add or remove fields before they are stored. You can choose to offload some, or all, of the data to a third-party storage system. You can choose whether to store any Analytics data in API Connect. Some options are configured during installation, while other options are easily changed after your Analytics subsystem is in service.

- [Counting total API calls for your Analytics service](#)  
Count the API calls for the IBM® API Connect Analytics subsystem and see the total categorized by response status.
- [Viewing API event data in API Connect](#)  
The gateway generates API event data for every API invocation and forwards this data to its associated analytics service. The captured event data can be viewed in the Cloud Manager UI, API Manager UI, from the toolkit CLI/REST API, or offloaded to third-party systems.
- [Including geoIP fields in analytics data](#)  
Configure analytics settings to include geoIP fields in the data.
- [Configuring data retention and index rollover time periods](#)  
You can set a time period for data retention and index rollover.
- [Configuring the Logstash persistent queue](#)  
The Logstash persistent queue is embedded in the data ingestion pipeline. It ensures that data collection continues in the event of a failure.
- [Offloading analytics to third-party systems](#)  
The event data that is generated and collected in your API Connect on-premises deployment can optionally be forwarded to external third-party systems for storage and analysis.
- [Applying filters to incoming analytics data](#)  
Update your analytics configuration to apply filters to incoming data. You can add, modify, and remove API event record contents before they are saved.

## Counting total API calls for your Analytics service

Count the API calls for the IBM® API Connect Analytics subsystem and see the total categorized by response status.

Note: If your IBM API Connect bases costs on usage, you can use the following methods to track this information so that you can accurately report it. If your organization did not deploy the Analytics subsystem, then this feature is not available. If you deployed Analytics but offload your data and do not store it in IBM API Connect, then you must capture usage information from the third-party application where you offload data.

You can see the total API calls over time for each analytics service in the consumption dashboard. To see the consumption dashboard, navigate to the Cloud Manager UI analytics view: [Cloud Manager analytics view](#).

Alternatively, the analytics CLI can be used to get your API count: [Using the Analytics CLI](#).

1. Use the following CLI command to retrieve a count of all Analytics events recorded by the Analytics service:

```
apic -m analytics events:cloudCount --server <management server platform api endpoint> --analytics-service <analytics service name> --format json
```


- *<management server platform api endpoint>* is the Management server platform api endpoint.
- *<analytics service name>* is the name of Analytics service as configured in the Cloud Manager.

2. If required, additional filters can be applied to the results. For example, `--timeframe last12hours`. See the `help` output from: `apic -m analytics events:cloudCount -h` for more details.

## Viewing API event data in API Connect

The gateway generates API event data for every API invocation and forwards this data to its associated analytics service. The captured event data can be viewed in the Cloud Manager UI, API Manager UI, from the toolkit CLI/REST API, or offloaded to third-party systems.

You can view API event data within API Connect in the following ways:

- The Analytics view of the Cloud Manager UI, which is accessed by clicking  in the navigation pane. API event data across all organizations, catalogs, and spaces is presented on *dashboards*. Each dashboard contains a set of related *visualizations* that depict analytics data as a graphical or metric representation, see [Cloud Manager analytics view](#)
- The Analytics view of the API Manager UI, which is accessed by clicking `Manage > Catalog_name > Analytics`. Analytics data for each Catalog or Space is collected into *dashboards*. Each dashboard contains a set of related *visualizations* that depict analytics data as a graphical or metric representation. Dashboards are provided for viewing analytics data for APIs, Plans, Products, Catalogs and Spaces. For more information about accessing and working with the API event data shown in API, Plan, and Product dashboards, see [Managing your Products](#).
- By using the debug function for the integrated test tool that is provided in the API Manager user interface.



The debug function is typically used to view information about the API configuration in order to identify possible causes for an API invocation failure.

- Using the toolkit CLI or REST API: [Using the analytics toolkit CLI](#)

API consumers who are members of a consumer organization can also view analytics data within dashboard views in the Developer Portal user interface. The dashboard views depict API invocation metrics about the APIs used either by a single application or within the entire consumer organization. For more information about accessing the analytics data for APIs that are invoked by developer applications, see [Analytics in the Developer Portal](#).

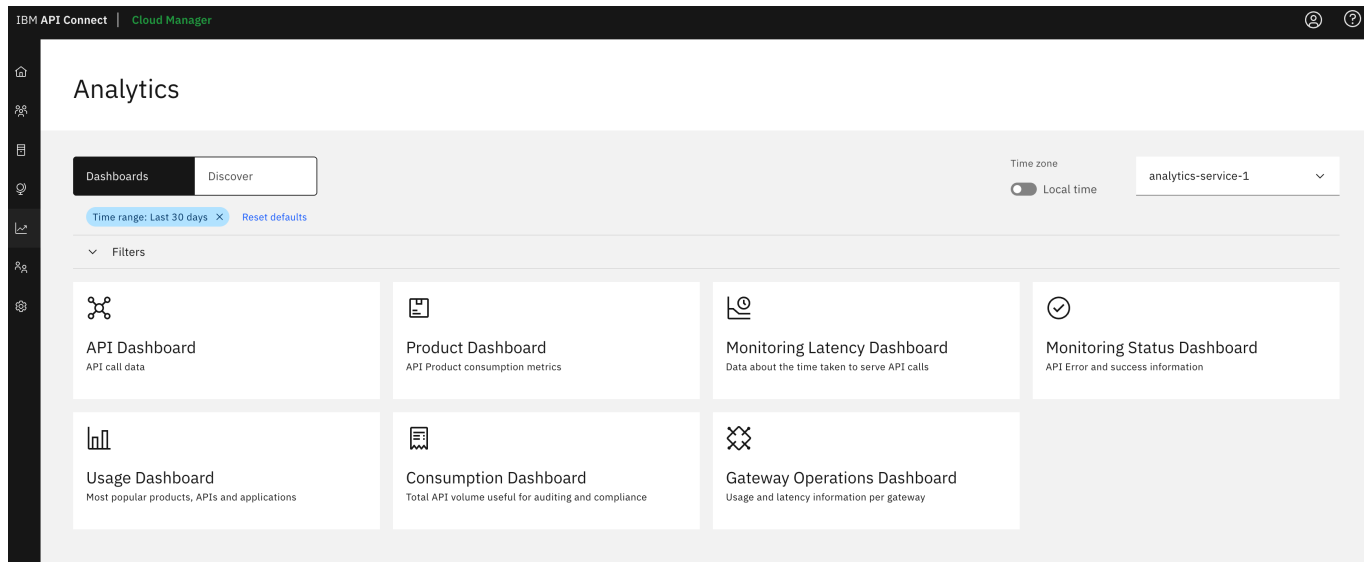
- [Cloud Manager analytics view](#)  
How to view your analytics event data in the Cloud Manager user interface.

## Cloud Manager analytics view

How to view your analytics event data in the Cloud Manager user interface.

API events from all catalogs, spaces, and organizations are captured and displayed together in the Cloud Manager analytics view. To view only analytics data for specific provider organizations, catalogs and spaces use the API Manager user interface [API Manager](#).

To access the analytics view click  in the navigation pane:

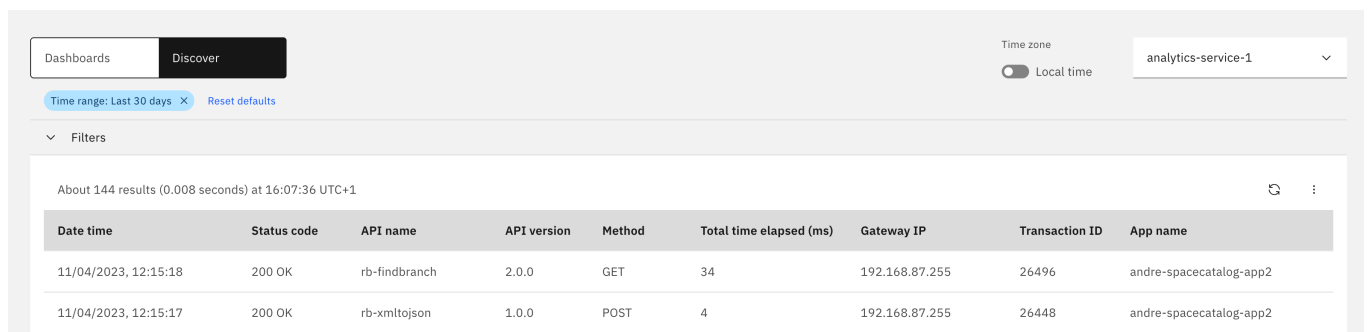


If you have multiple analytics services, use the drop-down at the upper right to select the analytics service whose API event data you want to view.

## Discover view

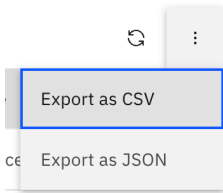
By default the Dashboards tab is displayed. To view your API event data in tabular form, select Discover:

### Analytics



The maximum number of API events that can be accessed from the UI is 10,000. The maximum number of event records that can be displayed per page on the Discover view is 200.

You can export the API event data from this view by clicking Actions at the upper right:



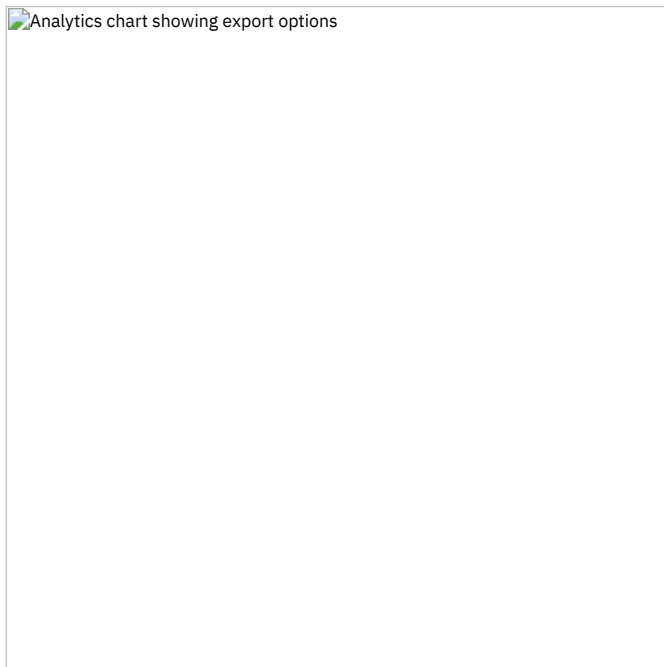
Note: When you export from the Discover view, only the data displayed on the current page of results is exported. If you want to export the data from all pages, you must export separately from each page. To export all your analytics data, do not use the UI, see [Exporting API event data](#).



## Dashboards view

Seven dashboards are available:

- API Dashboard- contains charts that summarize total API calls, response codes, and response times.
- Product Dashboard - contains charts that show total API calls and application subscriptions per plan.
- Monitoring Latency Dashboard - contains charts that provide response time statistics and data usage.
- Monitoring Status Dashboard - contains charts that show the response codes and success/failure rates of API calls.
- Usage Dashboard - contains charts that show the main five APIs, products, and apps.
- Consumption Dashboard - contains charts that show total API consumption in your cloud.
- Gateway Operations Dashboard - contains charts that show API usage and latency by gateway.

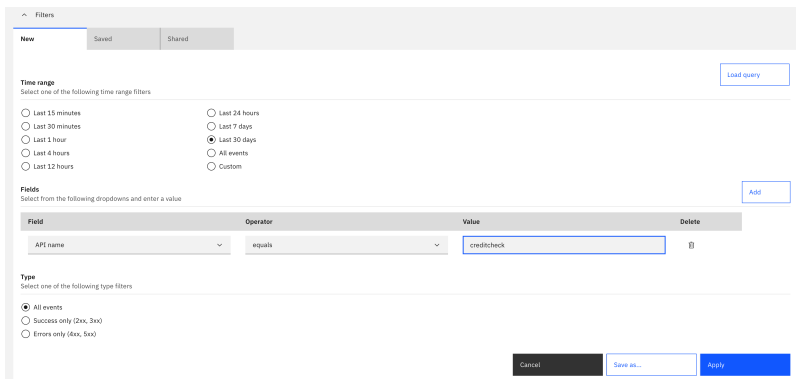
Each chart in the dashboards can be exported as a PNG or JPG image. The source data for each chart can be exported as a JSON or CSV file. To export a chart, click Actions :



It is also possible to enlarge the chart to full-screen by clicking  . To see the source data in tabular form click  .

## Filtering displayed data

You can filter the data that is displayed in the Dashboards and Discover view by defining queries. Expand the Filters section to view, define, and apply analytics queries to your displayed data:



You can save and share the queries that you define:

- Shared queries are visible to all users at the same scope. They are not visible to users in a different scope. For example, a query that is shared by a Cloud Manager UI user is not visible to an API Manager UI user. Within the API Manager UI, a query that is created at catalog scope is not visible to users viewing analytics data at provider organization scope.
- Other users cannot edit or delete queries that you share, but they can create a duplicate query, which they can edit and share.
- All queries have a name and an optional description. The name does not need to be unique.

Saved queries can be exported as strings for use in REST API and toolkit CLI queries, for example:



copies the following string to the clipboard:

```
api_name>equals:boo&timeframe=last30days
```

---

## Including geoIP fields in analytics data

Configure analytics settings to include geoIP fields in the data.

---

### About this task

By default, the `client_geoip` and `gateway_geoip` fields are not included in analytics data.

Configure the fields to include in your analytics data while deploying the analytics subsystem. For information on including the geoIP fields in analytics data, see the appropriate topic for your environment:

- Kubernetes: [Configure analytics to include geoIP fields.](#)
- OpenShift: [Configure analytics to include geoIP fields.](#)
- VMware: [Configure analytics to include geoIP fields.](#)

---

## Configuring data retention and index rollover time periods

You can set a time period for data retention and index rollover.

---

### About this task

Each API event is recorded as a document in a shared index. When this index reaches a maximum size or age it is archived, and a new index is created for recording new API events. This is known as index rollover. When an archived index reaches a maximum age, which is known as the data retention maximum, it is deleted. You can specify index rollover and data retention settings that control how often new indexes are created, and how long archived indexes are stored.

The index rollover settings determine how long an index accumulates data before it "rolls over" and a new index is created. You can control the duration of an index two ways: by setting a maximum age for the index (default is 1 day), and by setting a maximum number of documents that can be recorded in the index (default is 25 million documents). When the rollover occurs, a new index is created and new API events are recorded there, while the previous index is archived and kept until its age exceeds the data retention setting. If you change the rollover settings, consider both the amount of data to be stored within each index and the total number of indexes. Allowing indexes to grow too large, or storing too many indexes, can cause performance problems.

The data retention setting determines how long the stored indexes are retained. When an index's creation date exceeds the retention period, that index and all its data is deleted, even if some of the data that is stored in that index is younger than the retention period.

The default retention period is 30 days for a one replica deployment, and 90 days for a three replica deployment. Reasons for changing this setting include storage constraints, and data retention requirements for your organization. You might want to set this value to be less than the default if many API events are stored, especially if payload logging is enabled on the APIs.

The most recently created index (the index that is currently being written to) is not deleted, even if you set the retention period as small as 1 hour. If you regularly need to delete data quickly, adjust the Index Rollover setting so that a new index is rolled over to sooner, and the old index can be deleted.


Note: The rollover and retention settings have a significant impact on the memory that is required by the analytics subsystem. If you have rollover set to one day (1d), then to maintain an acceptable level of performance the following retention settings are recommended for each deployment profile:

- **n1xc2.m16:** This profile provides 3 GB of heap memory for storage on one node. Recommended retention is four days or less.
- **n1xc4.m32:** This profile provides 11 GB of heap memory for storage on one node. Recommended retention is 15 days or less.
- **n1xc6.m48:** This profile provides 19 GB of heap memory for storage on one node. Recommended retention is 25 days or less.
- **n3xc4.m16:** This profile provides 3 GB of heap memory for storage on three nodes. Recommended retention is 12 days or less.
- **n3xc4.m32:** This profile provides 11 GB of heap memory for storage on three nodes. Recommended retention is 45 days or less.
- **n3xc6.m48:** This profile provides 19 GB of heap memory for storage on three nodes. Recommended retention is 76 days or less.
- **n3xc8.m64:** This profile provides 27 GB of heap memory for storage on three nodes. Recommended retention is 108 days or less.

If rollover is set to higher than one day, then more analytics data can be stored before a performance deterioration is observed. However, a higher rollover setting has the disadvantage that more data is deleted when an index reaches the retention age. For example, if you set your rollover period to seven days and you have the default 90-day retention period, then when your oldest index reaches the 90 days since it was created, it is deleted and you lose your first seven days of event data. To view and update the rollover and retention settings, from v10.0.5.3 you can use the Cloud Manager UI. For previous releases, use the toolkit CLI.

## Procedure

---

1. From the Cloud Manager UI, click  Topology.
2. Select the Analytics Service you want to view or edit.
3. Click Advanced settings.
4. Modify your rollover and retention settings as required and click Save.

**v10.0.5.2 and previous releases:** Use the toolkit CLI.

5. View the current retention and rollover settings by using the `clustermgmt:getStorageSettings` analytics CLI command:

```
apic -m analytics clustermgmt:getStorageSettings --server [SERVER] --analytics-service [ANALYTICS SERVICE] --format json
```

- `<management server platform api endpoint>` is the Management server platform api endpoint.
- `<analytics service name>` is the name of Analytics service as configured in the Cloud Manager.

Example output:

```
{
 "rollover": {
 "min_doc_count": 25000000,
 "min_index_age": "1d"
 },
 "retention": {
 "min_index_age": "30d"
 }
}
```

"min\_" refers to the minimum age or number of documents that are reached before the action takes place.

6. To update the retention and rollover settings create a JSON file that is called `retention_and_rollover_settings.json`, paste in the output from step 5 and set the retention and rollover setting as required. Apply these new settings with: `clustermgmt:putStorageSettings`

```
apic -m analytics clustermgmt:putStorageSettings --server [SERVER] --analytics-service [ANALYTICS SERVICE] retention_and_rollover_settings.json
```

- `<management server platform api endpoint>` is the Management server platform api endpoint.
- `<analytics service name>` is the name of Analytics service as configured in the Cloud Manager.

---

## Configuring the Logstash persistent queue

The Logstash persistent queue is embedded in the data ingestion pipeline. It ensures that data collection continues in the event of a failure.

To enable and configure the persistent queue, see:

- [Kubernetes - Analytics persistent queue](#)
- [OpenShift - Analytics persistent queue](#)

After making changes to these settings the ingestion pod will restart.

---

## Offloading analytics to third-party systems

The event data that is generated and collected in your API Connect on-premises deployment can optionally be forwarded to external third-party systems for storage and analysis.

### About this task

---

API Connect supports the offload of API analytics data to the following third-party systems: HTTP, Elasticsearch v7, Apache Kafka, OpenSearch, and Syslog. You can configure a custom certificate for connecting to the third-party target and define filters to refine data that will be stored.

You typically configure data offloading while deploying the Analytics subsystem, but this configuration can be changed after installation, and involves updating the analytics custom resource and a restart of the ingestion pod.

- [Kubernetes - Offloading data to a third-party system](#)
- [OpenShift - Offloading data to a third-party system](#)
- [VMware - Offloading data to a third-party system](#)

- [Troubleshooting your analytics offload](#)

Review information on common issues with offloading Analytics data.

---

## Troubleshooting your analytics offload

Review information on common issues with offloading Analytics data.

If you determine that your third-party system is not receiving analytics data, you can complete some checks to help determine the cause and resolve the problem. For information on troubleshooting data offload, see the appropriate topic for your environment:

- Kubernetes, OpenShift, and Cloud Pak for Integration: [Troubleshooting your Analytics offload](#)

---

## Applying filters to incoming analytics data

Update your analytics configuration to apply filters to incoming data. You can add, modify, and remove API event record contents before they are saved.

To create and update filters, see:

- [Kubernetes - Modifying incoming analytics data](#).
- [OpenShift - Modifying incoming analytics data](#).

If you want to modify the data for specific APIs rather than all incoming data, you can use gateway scripts or the `set-variable` policy, as described here: [Modifying analytics API event data](#).

---

## Configuring audit logging to monitor user operations

Configure auditing in IBM® API Connect to monitor user operations such as who published which product and when.

Audit logging is useful when you want to monitor the create, read, update, delete, log-in, and log-out activity in your deployment; for example, who made changes, when, and what they changed.

Note: Actions performed by [Portal administrators](#) are not captured by the audit feature.

---

### What events are audited?

All create, update, delete, log-in, and log-out operations that are performed by users in the Admin UI, Manager UI, Toolkit, and REST API are captured by this feature. For Portal users, all create, update, and delete operations involving applications are captured.

Beginning in API Connect 10.0.5.3, read operations can also be audited when the `audit_reads` setting is additionally enabled (using the toolkit CLI).

Attention: If you choose to enable auditing of read operations for an HTTP endpoint, be aware of that the response might return large payloads, and might contain sensitive information.

Internal calls and calls between system components are not audited because they are part of the product functionality and are not executed by users.

---

### How are audit events logged?

Audit events are logged by sending the information about operations to one or more endpoints of the following types: HTTP, Syslog UDP, Syslog TCP, or Syslog TLS.

- HTTP endpoints:  
Audit logging messages are sent as HTTP/HTTPS requests and use the payload format defined by the Cloud Auditing Data Federation (CADF). If the endpoint requires authentication, you can configure a header that will be included in the audit logging request. Audit logging requests are issued once only, on the assumption that every endpoint is available to receive the requests. There is no error checking to ensure that requests are received and stored by the endpoints, and failed requests are not re-sent. If you suspect that audit logging requests are not successfully reaching an endpoint, you can check the server logs on the console, where any error messages are recorded.

There are two modes that control how the auditing is sent to an endpoint: blocking and non-blocking. In non-blocking mode (the default), an audit event is sent to the endpoint and the system does not wait for a response from the endpoint. Non-blocking mode provides the best performance. Blocking mode should be used with caution: when an audit event is sent, the system pauses and waits for a response from the endpoint. Blocking mode has a negative impact on performance.

- Syslog endpoints:  
Audit logging messages are sent as syslog messages. If the endpoint requires mutual TLS, you can configure a TLS profile to send the required client certificates. Audit logging requests are issued once only, on the assumption that every endpoint is available to receive the requests. There is no error checking to ensure that requests are received and stored by the endpoints, and failed requests are not re-sent. If you suspect that audit logging requests are not successfully reaching an endpoint, you can check the server logs on the console, where any error messages are recorded.

All syslog messages are sent as non-blocking regardless of the auditing mode, since syslog messages are designed to be fire and forget (syslog was originally a UDP protocol, which by its nature does not have any acknowledgment). Blocking is only used for HTTP endpoints.

Note: With the microservices nature of API Connect, any changes to the audit settings can take up to five minutes to be fully honored by all of the microservices supporting it.

---

### What's included in the audit logging request?

- HTTP endpoints:  
The payload containing the audit logging record includes information about the user who executed the operation, the date and time of the call, the resource affected by the call, the operation performed, and the payload. If you enabled audit logging for read operations, the payload might also contain the properties from the response (based on your configuration settings).

To see the most recent set of fields that are included in the payload, refer to the OpenAPI schema for the audit logging request in the Provider API, which you can download at [https://us-south.functions.cloud.ibm.com/api/v1/web/API%20Connect%20Native\\_apic-on-prem/default/v10-index.http](https://us-south.functions.cloud.ibm.com/api/v1/web/API%20Connect%20Native_apic-on-prem/default/v10-index.http).

HTTP example 1: sample payload for an audit logging message directed to an HTTP endpoint (IDs and users are not real):

```
{
 "id": "f6fcacb5-e8eb-4e2c-0f67-53b6a792d7f0",
 "typeURI": "http://schemas.dmtf.org/cloud/audit/1.0/event",
 "action": "update",
 "outcome": "success",
 "eventType": "activity",
 "reason": {
 "reasonCode": "200",
 "reasonType": "HTTP"
 },
 "eventTime": "2020-02-01T15:18:16.011Z",
 "initiator": {
 "id": "/api/user-registries/38385c9f-7837-583c-01f7-9d8c37a9a80d/10de0db0-27b5-35d7-b4b5-635eabb48b4a/users/24c6cddc-7a7e-5fc6-882d-5317d822048e",
 "name": "admin:default-idp-1/tjwatson",
 "typeURI": "service/security/account/user"
 },
 "target": {
 "id": "0beb6d21-6207-5381-b9a7-cc91a3e82c19",
 "typeURI": "tls_client_profile"
 },
 "observer": {
 "id": "target"
 },
 "requestPath": "/api/orgs/admin/tls-client-profiles/uma-tls/1.0.0",
 "requestData": {
 "url": "/api/orgs/admin/tls-client-profiles/uma-tls/1.0.0",
 "body": {
 "visibility": {
 "type": "public"
 },
 "namespace": "38385c9f-6726-472b-91f7-9d8c37a9a80d",
 "updated_at": "2020-02-01T15:18:15.924Z"
 }
 },
 "responseData": {
 "id": "0beb6d21-6207-5381-b9a7-cc91a3e82c19",
 "url": "https://my_host.example.com:3003/api/orgs/38385c9f-7837-583c-01f7-9d8c37a9a80d/tls-client-profiles/0beb6d21-6207-5381-b9a7-cc91a3e82c19",
 "name": "uma-tls",
 "version": "1.0.0",
 "title": "Uma TLS Client Profile"
 },
 "attachments": [
 {
 "timestamp": {
 "start": "2020-02-01T15:18:15.802Z",
 "end": "2020-02-01T15:18:16.011Z"
 },
 "cloud_name": "management.example.com",
 "request_id": "f6fcacb5-e8eb-4e2c-0f67-53b6a792d7f0",
 "method": "patch",
 "operation": "tls_client_profile_updateByNameVersion",
 "summary": "Update the TLS Client Profile object by name and version",
 "resource": "tls_client_profile",
 "user": {
 "url": "/api/user-registries/38385c9f-7837-583c-01f7-9d8c37a9a80d/10de0db0-27b5-35d7-b4b5-635eabb48b4a/users/24c6cddc-7a7e-5fc6-882d-5317d822048e",
 "context": "admin",
 "idp_name": "default-idp-1",
 "name": "tjwatson"
 },
 "registration": {
 "url": "/api/cloud/registrations/9d13c715-e3dc-55ef-92f1-1c2651c0a660",
 "type": "toolkit"
 }
 }
]
}
```

HTTP Example 2: sample of the `responseData` part of the message payload for a read operation (`GET /orgs`) with `list_results_limit=3`:

```
responseData:
 results_read:6
 total_results:6
 results:
 [0]:{ summary: "Administers the admin organization", org_url: "https://localhost:3003/api/orgs/18361c4c-54a2-41b2-9ff1-79f72832d405", updated_at: "2023-04-03T10:03:53.000Z", scope: "org", name: "administrator", created_at: "2023-04-03T10:03:53.000Z", id: "e42e304e-a335-4df0-876d-682cd8a36725", type: "role", title: "Administrator", api_version: "2.0.0", url: "https://localhost:3003/api/orgs/18361c4c-54a2-41b2-9ff1-79f72832d405/roles/e42e304e-a335-4df0-876d-682cd8a36725", permission_urls: [<cut>] }
 [1]:{ summary: "Minimum role", org_url: "https://localhost:3003/api/orgs/18361c4c-54a2-41b2-9ff1-79f72832d405", updated_at: "2023-04-03T10:03:53.000Z", scope: "org", name: "member", created_at: "2023-04-03T10:03:53.000Z", id: "d415fcd9-7e22-4522-af15-c62d3094599a", type: "role", title: "Member", api_version: "2.0.0", url: "https://localhost:3003/api/orgs/18361c4c-54a2-41b2-9ff1-79f72832d405/roles/d415fcd9-7e22-4522-af15-c62d3094599a", permission_urls: [<cut>] }
 [2]:{ summary: "Manages API provider organizations", org_url: "https://localhost:3003/api/orgs/18361c4c-54a2-41b2-9ff1-79f72832d405", updated_at: "2023-04-03T10:03:53.000Z", scope: "org", name: "organization-manager", created_at: "2023-04-03T10:03:53.000Z", id: "4c51446d-417d-4c4a-albe-083ecc699c2c", type: "role", title: "Organization Manager", api_version: "2.0.0", url: "https://localhost:3003/api/orgs/18361c4c-54a2-41b2-9ff1-79f72832d405/roles/4c51446d-417d-4c4a-albe-083ecc699c2c", permission_urls: [<cut>] }
```

- Syslog endpoints:

The nature of syslog messaging means that the information logged is less structured and shorter than information logged using structured HTTP endpoints. With syslog, the information is more of a summary.

The following sample response shows payloads for an audit logging message directed to a syslog endpoint (IDs and users are not real):

```
The user admin:default-idp-1/admin has deleted the resource Org 'alpha (Alpha title)', id 19cf81c7-7cb3-43de-b5e5-d92cdfb26214 and url https://localhost:3003/api/orgs/19cf81c7-7cb3-43de-b5e5-d92cdfb26214
The user admin:default-idp-1/admin has deleted the resource Gateway Service 'webhook-gw-v5 (Webhook Gateway Service (v5) title)', id 08b87e25-616d-451e-8766-bad1c7b82a42 and url https://localhost:3003/api/orgs/69c878db-4f11-4c3e-9499-10daa25f04de/availability-zones/cb2caa3d-d619-4c2e-85f8-709956495a7b/gateway-services/08b87e25-616d-451e-8766-bad1c7b82a42
The user provider:default-idp-2/steve@example.com has created the resource Catalog 'climbon:1.0.1 (The climbon product description)', id 665036df-28ad-447e-b8de-fe9d6967b66d and url https://localhost:3003/api/catalogs/19cf81c7-7cb3-43de-b5e5-d92cdfb26214/8641fda7-43f8-4e59-b405-58a4844016c9/products/665036df-28ad-447e-b8de-fe9d6967b66d
The user admin:default-idp-1/admin has read the resource Cloud Setting 'cloud-setting', id settings and url https://localhost:3003/api/cloud/settings
```

## Verifying that the endpoint is receiving audit logging messages

By default, auditing operates in `non_block` mode to ensure best performance. In `non_block` mode, each audit logging message is sent immediately without waiting to ensure that the previous message was received. If you suspect that audit logging messages are not being received by the endpoint, download the server logs from the Kubernetes console and examine the `apimicroservices` file for messages related to the connection state. Review the messages to validate that the connection was successful or to further diagnose issues.

For HTTP endpoints (not available for Syslog endpoints), you can test the connection by issuing a "test-connection" call using either the toolkit or a REST API. A response status of 204 indicates that the call was successful and the endpoint is available. The following example contains the payload from a connection test (IDs and users are not real):

```
{
 "id": "8d8eaaa2-1b46-4dcb-cadf-639d7d1f7f20",
 "typeURI": "http://schemas.dmtf.org/cloud/audit/1.0/event",
 "action": "create",
 "outcome": "success",
 "eventType": "activity",
 "reason": {
 "reasonCode": "200",
 "reasonType": "HTTP"
 },
 "eventTime": "2020-01-30T15:51:18.254Z",
 "initiator": {
 "name": "admin:default-idp-1/tjwatson",
 "typeURI": "service/security/account/user"
 },
 "target": {
 "id": "test-connection",
 "typeURI": "cloud_setting"
 },
 "observer": {
 "id": "target"
 },
 "requestPath": "/api/cloud/settings/audit-endpoint/test-connection",
 "requestData": {
 "url": "/api/cloud/settings/audit-endpoint/test-connection",
 "body": {
 "message": "hello"
 }
 },
 "responseData": {
 "name": "world"
 },
 "attachments": [
 {
 "timestamp": {
 "start": "2020-01-30T15:51:18.160Z",
 "end": "2020-01-30T15:51:18.254Z"
 },
 "cloud_name": "management.example.com",
 "request_id": "8d8eaaa2-1b46-4dcb-cadf-639d7d1f7f20",
 "method": "post",
 "operation": "cloud_setting_auditEndpointTestConnection",
 "summary": "Test connection using one of the auditing endpoints",
 "resource": "cloud_setting",
 "user": {
 "id": "/api/user-registries/dd49cebf-d3ba-5fee-92eb-f895328bc9ef/d51fa599-a5f3-50e2-a786-fafcc76daca5/users/7ff5fff8-3d4b-3524-8687-524740580d0b",
 "context": "admin",
 "idp_name": "default-idp-1",
 "name": "tjwatson"
 },
 "registration": {
 "url": "/api/cloud/registrations/734c04f7-0fe2-3ffa-b745-81027a3b8c76",
 "type": "toolkit"
 }
 }
]
}
```

Tip: To ensure that audit logging messages are not lost due to an unavailable endpoint, verify each endpoint periodically.

- [Configuring the audit settings](#)  
Use the Cloud Manager UI or the toolkit CLI to enable auditing and configure settings.

---

# Configuring the audit settings

Use the Cloud Manager UI or the toolkit CLI to enable auditing and configure settings.

## About this task

---

The audit feature tracks platform-level changes in API Connect through the use of platform APIs, CLI, and Web UI. For information on tracking run-time API execution, see [API Analytics](#).

You can configure audit settings using either the Cloud Manager UI, or the [Toolkit CLI](#).

## Using the Cloud Manager UI to configure audit settings

---

Use the Cloud Manager user interface to enable auditing of customer API calls.

### Before you begin


Configure one or more audit endpoints where logging data can be directed, and validate that every endpoint is available by testing the connection as explained in [Configuring audit logging to monitor user operations](#).

### About this task

By default, auditing is disabled. If you enable auditing, the API Connect management node emits audit events as messages and sends them to the endpoint for storage.

Note: Auditing for read operations can only be enabled and configured [using the CLI](#).

### Procedure

1. In the Cloud Manager, click  Settings > Audit Setting > Edit.
2. Click Enable to enable the audit feature, or Disable to disable it.
3. Select an audit mode:
  - Non-blocking - Default value, offers the best performance. Each request is passed immediately without waiting for a response to the previous request. This is susceptible to errors if a request is dependent on the results of an earlier request whose response is not received before the new request is sent.
  - Blocking - If you select this mode, expect a degradation in performance. Each request is allowed to finish execution before the next request is sent. This mode results in fewer errors when requests are dependent on the responses of earlier operations, but the delayed requests slow the network.  
Note: Blocking mode is only applied to HTTP endpoints and is ignored for Syslog endpoints.
4. Click Add and select the type of endpoint that you want to add (you must add at least one endpoint). Endpoints can be of the following types: HTTP, Syslog UDP, Syslog TCP, or Syslog TLS.
  5. In the Endpoints section, fill in the fields to describe the new endpoint:
    - HTTP**
      - URL - Required, must not be empty. The URL of the server where audit logging messages will be sent.  
If you previously saved the audit settings with an empty URL and experienced problems, you can edit the URL to correct it.
      - TLS client profile - Optional. The URL of the server where the endpoint's TLS client profile is stored.
      - Header name and Header value - Optional. Multiple headers are allowed (click Add to add another header). For each header, specify the header name and the header value.
    - Syslog UDP**
      - Host name - Required, must not be empty. The host name of the server where audit logging messages will be sent; for example: www.example.com. Do not include the `syslog://` protocol with the host name. When audit logging messages are directed to the host server, the protocol is added to the host name programmatically.
      - Port - Optional. The port number where audit logging messages should be directed.
    - Syslog TCP**
      - Host name - Required, must not be empty. The host name of the server where audit logging messages will be sent; for example: www.example.com. Do not include the `syslog://` protocol with the host name. When audit logging messages are directed to the host server, the protocol is added to the host name programmatically.
      - Port - Optional. The port number where audit logging messages should be directed.
    - Syslog TLS**
      - Host name - Required, must not be empty. The host name of the server where audit logging messages will be sent; for example: www.example.com. Do not include the `syslog://` protocol with the host name. When audit logging messages are directed to the host server, the protocol is added to the host name programmatically.
      - Port - Optional. The port number where audit logging messages should be directed.
      - TLS client profile - Optional. The URL of the server where the endpoint's TLS client profile is stored.
  6. Verify that you can reach the endpoint by clicking Test.
  7. To specify another endpoint, click Add at the beginning of the "Endpoints" section.
  8. When you finish adding endpoints, click Save to save your settings.

### Results

With the microservices nature of API Connect, any changes to the audit settings can take up to five minutes to be fully honored by all of the microservices supporting it.

### What to do next

If you suspect that audit logging messages are not being received by the endpoint, see [Verifying that the endpoint is receiving audit logging messages](#) for information on validating the connection.



## Using the CLI to configure audit settings

Use the IBM API Connect toolkit CLI to enable auditing of API calls.

### Before you begin

Configure one or more audit endpoints where logging data can be directed, and validate that every endpoint is available by testing the connection as explained in [Configuring audit logging to monitor user operations](#).

### About this task

By default, auditing is disabled. If you enable auditing, the API Connect management node emits audit events as messages and sends them to the endpoint for storage.

Use the following `audit_setting` options to enable auditing with the CLI toolkit:

- `enable` - Boolean, defaults to `false` (no audit). Set to `true` if you want to enable auditing.
- `mode` - Enumerated, defaults to `non_block`. The audit mode determines how audit requests are timed when they are sent to the endpoint:
  - `non_block` - Default value, offers the best performance. Each request is passed immediately without waiting for a response to the previous request. This is susceptible to errors if a request is dependent on the results of an earlier request whose response is not received before the new request is sent.
  - `block` - If you select this mode, expect a degradation in performance. Each request is allowed to finish execution before the next request is sent. This mode results in fewer errors when requests are dependent on the responses of earlier operations, but the delayed requests slow the network.  
Note: Block mode is only applied to HTTP endpoints and is ignored for Syslog endpoints.
- `audit_reads` - Use this setting to enable auditing of read operations:
  - `enable` - Boolean, defaults to `false` and no audit events are sent to the endpoint for read operations. Set to `true` if you want to enable audit logging for read operations.
  - `response_data_to_http_endpoints` - Configure additional settings for HTTP endpoints only (the settings do not apply to Syslog endpoints).
    - `enable` - Boolean, defaults to `false` and audit events for GET operations are sent to the HTTP endpoint with no response data in the event. Set to `true` if you want to include the response data.  
Attention: If you choose to include the response data, be aware that the response might return large payloads, and might contain sensitive information.
    - `list_results_limit` - Integer, must be 0 or larger. Use this setting to specify the maximum number of results of `GET list` operations to include in the audit message (default is 0 results).  
  
If `audit_reads` is enabled, then the audit message for a `GET` of a *single object* (for example, `GET /orgs/admin`) always includes all of the properties of the response object.  
  
If the `list_results_limit` is set to 0, then the audit message for a `GET` to *list objects* (for example, `GET /orgs`) does not include any objects from the GET response. If you set the `list_results_limit` to a value greater than 0, then the audit message includes all properties from the first `list_results_limit` number of objects from the `GET /orgs` response.
- `endpoints` - Array of strings. You must specify at least one endpoint where the audit logging messages are sent. Endpoints can be of the following types: HTTP, Syslog UDP, Syslog TCP, or Syslog TLS.  
For HTTP and Syslog TLS endpoints, you can optionally include a TLS client profile URL. For HTTP endpoints, you can additionally include one or more optional headers. For example, if the endpoint requires an authorization header, you can include it in the endpoint setting.
  - `endpoint_type` - String. Required. Allowed values are: `http`, `syslog_udp`, `syslog_tcp` or `syslog_tls`.
  - `endpoint` - String. Required, must not be empty or blank. The URL of the server where audit logging messages will be sent. Include the protocol (`http://` or `syslog://`) and the port, if applicable. For example, `syslog://<hostname>:<port>`.
  - `tls_client_profile_url` - String. Optional; supported only for HTTP and Syslog TLS endpoints. The URL of the server where the endpoint's TLS client profile is stored.
  - `headers` - String. Optional; supported only for HTTP endpoints. Multiple headers are allowed. For each header, specify the header name and the header value using the format shown in the example.

The list of audit properties can be lengthy, so uploading the settings in a file helps to avoid errors. Format the settings as a `.yaml` file (you can choose the file name) and upload the file using the toolkit CLI. The following example enables auditing in non-blocking mode with multiple endpoints to demonstrate how the settings should be formatted.

```
audit_setting:
 enabled: true
 mode: non_block
 audit_reads:
 enabled: true
 response_data_to_http_endpoints:
 enabled: true
 list_results_limit: 0
 endpoints:
 - endpoint_type: syslog_udp
 endpoint:
 endpoint: syslog://syslog-audit-endpoint.example_1.com:514
 - endpoint_type: syslog_tls
 endpoint:
 endpoint: syslog://syslog-tls-audit-endpoint.example_1.com:1601
 tls_client_profile_url: https://security.tls_example.com/api/orgs/d8eac736-b4e2-3aa6-c7ff-978a4cd19d0b/tls-client-profiles/19d0f087-b4e2-3aa6-926b-9f6d51d028c3
 - endpoint_type: http
 endpoint:
 endpoint: https://mutt-audit-endpoint.example_1.com/auditing
 - endpoint_type: http
 endpoint:
 endpoint: https://auditing-endpoint.example_1.com/inputs/64e38c02-acd9-6530-76f8-dd0f336d7b11
 tls_client_profile_url: https://security.tls_example.com/api/orgs/d8eac736-b4e2-3aa6-c7ff-978a4cd19d0b/tls-client-profiles/19d0f087-b4e2-3aa6-926b-9f6d51d028c3
```

```
headers:
 authorization: Basic c3Bvb246c2ltb25rYXRpZlZlYQ==
- endpoint_type: http
 endpoint:
 endpoint: https://auditing-endpoint.example_2.com/inputs/64e38c02-acd9-6530-76f8-dd0f336d7b11
 headers:
 authorization: Basic c3Bvb246c2ltb24=
 x-header: ForYourEyeOnly
```

## Procedure

1. Create a YAML file with the audit settings as shown in the example.

For information on using input files with the toolkit CLI, see [Reading input from the command line](#).

2. Run the following command to log in to a management server as a member of the cloud administration organization:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm admin/identity_provider
```

Cloud settings are stored on a management server and you must log in to the server before you can access the settings with the CLI.

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`. For more information on logging in, see [Logging in to a management server](#).

3. Upload the YAML file with the auditing settings by running the following command:

```
apic cloud-settings:update --server mgmt_endpoint_url my_audit_settings.yaml
```

4. Run the following command to log out of the management server:

```
apic logout --server mgmt_endpoint_url
```

## Results

With the microservices nature of API Connect, any changes to the audit settings can take up to five minutes to be fully honored by all of the microservices supporting it.

## What to do next

If you suspect that audit logging messages are not being received by the endpoint, see [Verifying that the endpoint is receiving audit logging messages](#) for information on validating the connection.



---

## Reviewing a gateway's processing status

Review the processing status for the Gateway services that are enabled in your API Connect Catalogs and Spaces.

### About this task

Sometimes you might want to check the status of an API event that is being processed by the gateway. For example, you might want to ensure that a Product was published before and is available on the gateway before you invoke one of its APIs. Or perhaps it appears that API events are being processed too slowly and you want to see if all of the events for a particular gateway service are affected.

You can review a gateway's processing status in the API Manager user interface, or by [using the toolkit CLI](#) to retrieve the information.

Notes:

- You can only obtain processing status information for the DataPower® API Gateway. This feature is not supported for the DataPower Gateway (v5 compatible) service.
- Processing information is collected for each Catalog. If you view information for a particular Space, the display shows the results for the entire Catalog instead.

You can view the gateway processing status using the [UI](#) or the [CLI](#).

---


## Using the UI to view the gateway processing status

### About this task




Use the API Manager user interface to view the processing status for all of the gateway services associated with a particular Catalog or Space.

## Procedure


1. Open the Gateway services page:

- a. Log in to API Manager.
- b. In the navigation pane, click  Manage.
- c. Select a Catalog.
- d. Click the Catalog settings tab.
- e. On the Settings page, click Gateway services.

A list of all of the gateway services for the selected Catalog, including all of its Spaces, displays the gateway type and its endpoint URL. In addition, a status icon displays for each service name to provide a quick indication of that service's current status:

-  Normal: Between 0 and 10 events are waiting to be processed
-  Increased: Between 10 and 20 events are waiting to be processed
-  High: More than 20 events are waiting to be processed

You can view details about a service's status to gain insight into possible issues.

2. On the Gateway services page, click  next to a service and select View gateway processing status.

The Gateway processing status page displays a list of the events that were sent to the gateway. For each event, you can see the following information:

- Event type: The action performed by the event; for example, subscribing to a product, deleting an application, or creating a new consumer organization.
- Category: The feature area affected by the event; for example, subscriptions, applications, or consumer organizations.
- Time: The time when the event was processed by the gateway service.
- Status: The current event's processing status; for example, Queued (not yet sent for processing), Sent (but not processed yet), or Processed (complete).  
The number of events in the Queued and Sent states determine the status level for the service.

The display includes events that were not processed yet, as well as the last event processed. If a particular event is not listed as Sent or Queued, then that event was already processed.

## Using the CLI to obtain a gateway's processing status

### About this task

You can use the developer toolkit CLI to obtain information about the processing status for the Gateway services that are enabled in your Catalogs and Spaces.

The processing status information is returned in the following fields:

Field	Description
<code>service_up_to_date</code>	If <code>true</code> then all outstanding events have been processed by the Gateway service.
<code>service_state</code>	An overall status summary of the various states of events and processing by the Gateway service: <ul style="list-style-type: none"><li>• <code>green</code> indicates normal status with between 0 and 10 events waiting to be processed</li><li>• <code>orange</code> indicates that between 10 and 20 events are waiting to be processed</li><li>• <code>red</code> indicates that more than 20 events are waiting to be processed</li></ul>
<code>last_processed_event</code>	Details of the last event that was processed by the Gateway service, helping to provide a timeline of when the last event was processed, and some statistics about how long ago it was and how long the processing took.
<code>last_sent_event</code>	Details of the last event that was sent to the gateway service
<code>number_of_outstanding_sent_events</code>	The number of events that have been sent but not yet processed by the Gateway service.
<code>number_of_outstanding_queued_events</code>	The number of events that have been queued but not yet sent to the Gateway service.

The following example shows a "green" response that includes the status information and the list of sent events:

```
gateway_processing_status:
 service_up_to_date: false
 service_state: green
 last_processed_event:
 event_id: 40e1a564-2aea-429b-9d35-f588218688a1
 title: None provided
 processed_at: '2020-06-02 22:40:04.671+00'
 generated_at: '2020-06-02 22:39:33.856769+00'
 filter: product_replace
 event_processing_time: 24526
 elapsed_time_since_last_processed_event: 13682
 last_sent_event:
 event_id: 3bb52860-29f8-41a9-93d0-c9465a0a30e0
 title: None provided
 sent_at: '2020-06-02T22:40:15.820Z'
 generated_at: '2020-06-02 22:40:12.980079+00'
 number_of_outstanding_sent_events: 9
 number_of_outstanding_queued_events: 0
events:
 sent_events:
 - title: >-
 Update Product (name-version:lts-cat0-product-14:1.0.0, id:
 0df5bf4b-6706-4c36-aa0c-9c8ba2f15254) by user shavon0
 filter: product_update
 generated_at: '2020-06-02 22:39:45.430744+00'
 - title: >-
 Update Product (name-version:lts-cat0-product-64:1.0.0, id:
 4795106e-787d-41e6-bb94-9e6506719529) by user shavon0
 filter: product_lifecycle
 generated_at: '2020-06-02 22:39:46.11499+00'
 - title: None provided
```

```

filter: product_replace
generated_at: '2020-06-02 22:39:46.862433+00'
- title: >-
 Update Product (name-version:lts-cat0-product-15:1.0.0, id:
 6c7bf5ef-dca4-4b63-8060-2cc3c4d91d3b) by user shavon0
filter: product_update
generated_at: '2020-06-02 22:39:58.397278+00'
- title: >-
 Update Product (name-version:lts-cat0-product-65:1.0.0, id:
 4019df05-27a6-4ac2-83c8-6dd5da100105) by user shavon0
filter: product_lifecycle
generated_at: '2020-06-02 22:39:59.02833+00'
- title: None provided
filter: product_replace
generated_at: '2020-06-02 22:39:59.740152+00'
- title: >-
 Update Product (name-version:lts-cat0-product-16:1.0.0, id:
 4435a953-3c8d-4814-ada8-c71409cabcaf) by user shavon0
filter: product_update
generated_at: '2020-06-02 22:40:11.571056+00'
- title: >-
 Update Product (name-version:lts-cat0-product-66:1.0.0, id:
 fb9239d5-6064-4500-9a7b-63b841f5321a) by user shavon0
filter: product_lifecycle
generated_at: '2020-06-02 22:40:12.199838+00'
- title: None provided
filter: product_replace
generated_at: '2020-06-02 22:40:12.980079+00'
queued_events: []

```

The following example shows a "red" response:

```

gateway_processing_status:
service_up_to_date: false
service_state: red
last_processed_event:
event_id: 07387d98-dbfc-494b-af12-d7db91208e4c
title: >-
 Update Product (name-version:lts-cat0-product-49:1.0.0, id:
 25eba540-0cb1-4808-be3d-f22f65540cb2) by user shavon0
processed_at: '2020-06-02 22:32:10.838+00'
generated_at: '2020-06-02 22:31:43.38615+00'
filter: product_del
event_processing_time: 22597
elapsed_time_since_last_processed_event: 18427
last_sent_event:
event_id: 93d0bf8f-ebfb-4a8b-aa22-7f805719d5c8
title: >-
 Update Product (name-version:lts-cat0-product-11:1.0.0, id:
 d0bf799f-7f3a-4546-b267-a85c6a7f4b10) by user shavon0
sent_at: '2020-06-02 22:32:28.426+00'
generated_at: '2020-06-02 22:32:25.505109+00'
number_of_outstanding_sent_events: 37
number_of_outstanding_queued_events: 1

```

The result includes events that were not processed yet, as well as the last event processed. If a particular event is not listed as Sent or Queued, then that event was already processed.

## Procedure

To obtain processing status information for a Gateway service, complete the following steps:

1. Log in to your API Connect Management server as a provider organization member, by using the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

where:

- *mgmt\_endpoint\_url* is the platform API endpoint URL.
- *user\_ID* is the user ID of your provider organization account, and is the same as the user ID that you use to log in to the API Manager user interface.
- *password* is the password for your administrator account.
- *identity\_provider* is the identity provider that is used to authenticate API Manager users.

For example:

```
apic login --server platform-api.myserver.com --username myuser --password password --realm provider/myldap
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```

apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap

```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at

creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`. For full details on how to log in to your management server from the CLI, see [Logging in to the management server](#).

2. Obtain the Gateway processing status for a Gateway service that is enabled in a Catalog by running the following command:

```
apic configured-gateway-services:get --server mgmt_endpoint_url --scope catalog --org organization_name --catalog catalog_name gateway_service_name --fields "add(gateway_processing_status)" --output -
```

where:

- `mgmt_endpoint_url` is the platform API endpoint URL.
- The `--scope scope` parameter specifies `catalog` as the scope of the request.
- `organization_name` is the name of the provider organization that contains the Catalog.
- `catalog_name` is the name of the Catalog in which the Gateway service is enabled.
- `gateway_service_name` is the name of the Gateway service whose processing status you want to obtain.
- The `--fields "add(arguments)"` parameter can include either, or both, of the following arguments (separated with a comma):
  - `gateway_processing_status`: returns the total number of sent events, queued events, and processed events
  - `events`: returns lists of sent events with details about each event

For example:

```
apic configured-gateway-services:get --server platform-api.myserver.com --scope catalog --org myorg --catalog sandbox my-gateway-service --fields "add(gateway_processing_status)" --output -
```

---


## Changing your Cloud Manager password and profile information

You can change your Cloud Manager password and update your profile information.

### Procedure

---

To change your password or profile information, follow these steps:

1. Log in to the Cloud Manager user interface.
2. Click the User icon , and then select My Account.
3. In the Profile section, enter a new email address and First Name and Last Name for the account owner.
4. Click Save.
5. In the Change password section, enter your current password and your new password, and confirm the new password. The password must contain six characters at a minimum. It must contain at least three of the following types of characters:
  - Lowercase letters
  - Uppercase letters
  - Numbers
  - Characters from the following list of allowed punctuation and special characters:

```
!
\"
#
$
%
&
\
(
)
*
+
, (comma)
- (dash/hyphen)
. (period)
/
: (colon)
; (semi-colon)
<
=
>
?
@
[
\\
]
^
_ (underscore)
` (back quote/grave accent)
{
| (pipe)
}
~ (tilde)
```

The password cannot contain more than two consecutive repeating characters.

6. Click Save.

---

### Results

Your profile information and password are changed.

## Related concepts

- [Configuring and managing your server environment](#)

---

## Resolving login problems by increasing HTTP header size

You can resolve login problems for the Cloud Manager UI by increasing the maximum HTTP client header size.

### About this task

Login attempts to the user interface might fail with error 502 (**Bad Gateway**). This error can occur when logging in from a browser that has a large number of cookies. The error occurs because size of the login request exceeds the maximum HTTP client header size.

The maximum HTTP client header size is limited for security reasons. To workaroud this issue, you can clear the browser cache and cookies, or open an incognito window from the browser, and then retry the login.

Alternatively, you can increase the maximum HTTP client header size. Use caution when increasing the maximum size because larger headers can raise a security risk to your system.

### Procedure

1. **SSH** to your management system or appliance.
2. Enter the following commands:

```
kubectl get deployment -n namespace | grep apim-v2
kubectl edit deployment -n namespace apiconnect-apim-v2-deployment
```

3. In the **env** section of the deployment configuration, change `--max-http-header-size=12000` to a larger value that works for your environment. For example:

```
- name: NODE_OPTIONS
 value: --max-old-space-size=8094 --max-http-header-size=16000
```

4. Save the updated configuration.
5. Run `kubectl get pod -n namespace` a few times until the new `apiconnect-apim-v2` pod is up and running.

---

## Onboarding a new admin for Cloud Pak for Integration

Use the API Connect toolkit CLI to create a new admin account for use with Cloud Pak for Integration.

### About this task

If you changed the local admin account for Cloud Pak for Integration, you must create a new admin user in API Connect to ensure that the new Cloud Pak for Integration admin can log in to API Connect with the Common Services User Registry.

### Procedure

1. Download and install the API Connect toolkit CLI and credentials file as explained in [Installing the toolkit](#).
2. Log in to the toolkit CLI.  
Logging in to the toolkit CLI requires the endpoint URL for your API Connect management server. In Cloud Pak for Integration, the URL indicates the location of the management server within the Cloud Pak for Integration deployment.

- a. Determine the URL to the API Connect management server by running the following command:

```
oc -n APIC_namespace get mgmt APIC_instance -o jsonpath="{.status.zenRoute}" && echo ""
```

The response looks like the following example URL:

```
apic-mgmt-admin-apic.deve-cip-hlag-8e86d3798137c73f524cf9-0000.eu-de.containers.appdomain.cloud
```

- b. Log in to the API Connect toolkit CLI by running the following command:

```
apic -s API_Connect_URL login -u admin -p <password> -r admin/default-idp-1
```

where `API_Connect_URL` is the URL of the management server in your API Connect instance.

3. Create a new admin account by completing the following steps.  
Use the toolkit CLI to create an admin account in the Common Services User Registry and then add the new admin to the API Connect **administrator** role.

- a. Create a YAML file called `newCp4iAdmin.yaml` with the following information for the new admin account.

```
username: <new_cp4i_admin_username>
first_name: CP4I
last_name: Administrator
```

- b. Create the CP4i admin account by running the following command:

```
apic -s API_Connect_URL users:create -o admin --user-registry common-services newCp4iAdmin.yaml
```

c. Verify that the account was created by running the following command:

```
apic -s API_Connect_URL users:get -o admin --user-registry common-services <new_cp4i_admin_username> --output -
```

d. Get the new admin account's URL by running the following command:

```
apic -s API_Connect_URL users:list -o admin --user-registry common-services | grep <new_cp4i_admin_username> | awk '{print $4}'
```

e. Get the URL of the API Connect **administrator** role by running the following command:

```
apic -s API_Connect_URL roles:list --scope org -o admin | grep -w administrator | grep -v topology | awk '{print $2}'
```

f. Create a YAML file called newcp4iadminmember.yaml with the following content:

```
name: <new_cp4i_admin_username>
user:
 url: <user_URL>
role_urls:
 - <role_URL>
```

g. Use the newcp4iadminmember.yaml file to add the new admin as a member of the API Connect instance's admin organization. Create the new member by running the following command:

```
apic -s API_Connect_URL members:create --scope org -o admin newcp4iadminmember.yaml
```

h. Verify that the new admin account works by using it to log in to the Cloud Manager interface in your API Connect instance.

4. Remove the old Cloud Pak for Integration admin user from API Connect.

After you verify that the new Cloud Pak for Integration admin can log into the API Connect Cloud Manager interface, delete the previous Cloud Pak for Integration admin account.

a. Remove the previous admin from the membership of the API Connect admin organization by running the following command:

```
apic -s API_Connect_URL members:delete --scope org -o admin cs-admin
```

When you initially deployed API Connect as a component of Cloud Pak for Integration, the admin username was **admin** and the member in the API Connect admin organization was **cs-admin**. If the **cs-admin** member does not exist in the admin organization, then the Cloud Pak for Integration admin user was changed after deployment. In this case, the member uses the same name as admin account's user name. Delete the member with that name:

```
apic -s API_Connect_URL members:delete --scope org -o admin <old_cp4i_admin_username>
```

b. Delete the previous Cloud Pak for Integration admin user account from the Common Services User Registry by running the following command:

```
apic -s API_Connect_URL users:delete -o admin --user-registry common-services <old_cp4i_admin_username>
```

---

## Extending the Gateway server behavior

To support your enterprise requirements, you can extend the Gateway servers within IBM® API Connect to provide extra enforcement behavior.

---

### Before you begin

Before you develop your extensions, consider the guidelines in the following topics depending on which type of gateway you are using:

- [Gateway extension guidelines - DataPower Gateway \(v5 compatible\)](#)
- [Gateway extension guidelines - DataPower API Gateway](#)

For information on the different types of gateway, see [API Connect gateway types](#).

---

### About this task

API Connect Gateway servers uses a subset of DataPower® enforcement capabilities. You can supply DataPower Extensions to customize these enforcement capabilities further.

The DataPower extensibility function can be used to perform actions that include schema validation, antivirus scanning, message filtering, authentication, and authorization, token translation, message enrichment, encryption & decryption, digital signing, and validation and message transformation. For more information, see [IBM DataPower documentation](#).

Important: Gateway server extensions must not be used to modify objects that cause the **apic-gw-service** object to restart, such as TLS Profiles, and Gateway Peering. Extensions that cause the **apic-gw-service** object to restart, lead to unpredictable behavior in the Gateway service.

---

### Procedure

To extend the default enforcement capabilities that are provided in API Connect for the Gateway server, complete the following steps.

1. In your IBM DataPower environment, develop the configuration that you want to add to your Gateway server.
2. Test your enforcement configuration before you add the configuration to the Gateway server.
3. When you complete the IBM DataPower configuration, save the enforcement objects and files as a DataPower exported .zip file. The package file is ready to be uploaded to the Gateway server.
4. Copy your exported configuration .zip file to a centralized file system ready for upload to the Gateway server.

---

### What to do next

Upload your extension to the Gateway server; see [Configuring your Gateway server extensions](#).

- **Gateway extension guidelines - DataPower Gateway (v5 compatible)**  
Before you develop your extensions to the DataPower Gateway (v5 compatible), consider these guidelines.
- **Gateway extension guidelines - DataPower API Gateway**  
Before you develop your extensions to the DataPower API Gateway, consider these guidelines.
- **Configuring your Gateway server extensions**  
You can add extra IBM DataPower enforcement capabilities to a Gateway service by uploading a .zip file that defines the required extension behavior, and then enabling the extension in IBM DataPower.

DataPower Gateway (v5 compatible)

## Gateway extension guidelines - DataPower Gateway (v5 compatible)

Before you develop your extensions to the DataPower® Gateway (v5 compatible), consider these guidelines.

Note: These guidelines apply to the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway, see [Gateway extension guidelines - DataPower API Gateway](#).

For information on the different types of gateway, see [API Connect gateway types](#).

You can download sample gateway extensions from <https://github.com/ibm-apiconnect/dp-extensions>.

You can create DataPower Processing Rules that can extend the enforcement behavior of the IBM® API Connect Gateway server at the following locations:

- **pre-request** extension:
  - Before the Gateway server begins to use the policies in the assembly to process the request.
- **post-request** extension:
  - After the Gateway server processes all of the policies in the assembly up to the proxy policy, if a proxy policy is used.
  - After the Gateway server processes all policies in the assembly, but before any catch logic is processed, if a proxy policy is not used.
- **post-response** extension:
  - After the Gateway server processes all of the remaining policies in the assembly after the proxy policy (including catch logic), but before the response is returned to the client application, if a proxy policy is used.
  - After the Gateway server processes the catch logic, but before the response is returned to the client application, if a proxy policy is not used.
- **post-error** extension:
  - If an error occurs, then before the Gateway server returns the error response to the client application.

To configure the Gateway server to call your extension Processing Rule, you must create an XML file that indicates the extension location and Processing Rule name. For example,

```
<extensions>
<extension location="pre-request">CustomRule1</extension>
<extension location="post-request">CustomRule2</extension>
<extension location="post-response">CustomRule3</extension>
<extension location="post-error">CustomRule4</extension>
</extensions>
```

The `<extension>` element entries are optional for any of the locations. Refer to the Gateway server Extension schema for the XML file syntax.

An extension Processing Rule can be applied to a specific organization or to all of the organizations and Catalogs. If the `<extension>` element does not have a tenant attribute, then the `<extension>` element is applied to all of the organizations. The following example shows an `<extension>` element without a tenant attribute.

```
<extension location="post-error">gen_error_handling</extension>
```

To apply an extension Processing Rule to a specific organization, enter the organization name as the value for the tenant attribute in the `<extension>` element. The following example shows an `<extension>` element with a tenant attribute value of `organization1`.

```
<extension location="post-error" tenant="organization1">organization1_error_handling</extension>
```

If you want to exclude a specific organization in an extension Processing Rule, enter the organization name as the value for the tenant attribute in an empty `<extension>` element. Add the empty `<extension>` element immediately after the custom rule `<extension>` element that you want to exclude the organization from. The following example shows an empty `<extension>` element with a tenant attribute value of `organization1`.

```
<extension location="pre-request">a_specified_CustomRule</extension>
<extension location="pre-request" tenant="organization1"></extension>
```

Important: An extension Processing Rule with a specified tenant attribute takes precedence over an `<extension>` element that applies to all of the organizations. Also, only one extension Processing Rule is applied to each extension location.

This XML file must be saved to the following location and included in your DataPower exported configuration .zip file:

```
local:///ext/extensions.xml
```

There can be only one DataPower exported configuration .zip file added to a Gateway server in API Connect.

CAUTION:

- Do not modify these files directly in the DataPower API Gateway file system, use the gateway extension mechanism described on this page.
- Do not apply gateway extension modifications to configuration in the local:/config-sequence.cfg file.
- Do not use Gateway server extensions to modify objects that cause the `apic-gw-service` object to restart, such as TLS Profiles, and Gateway Peering. Extensions that cause the `apic-gw-service` object to restart, lead to unpredictable behavior in the Gateway service.

## DataPower configuration restrictions

All Processing Rules have read access to the INPUT context.



Processing Rules must not rely on context variables that are created by the IBM Gateway server enforcement configuration, because those configuration variables might change in the future.

All Processing Rules except Before Request can transform or alter the message flowing through the Gateway server. Ensure that the Processing Rule returns the desired message output context back to the Gateway server at the end of processing.

To avoid name conflicts, all DataPower configuration object names prefixed with **webapi** are reserved for IBM use.

The following folders cannot be modified:

- `local:///isp/*`
- `local:///gwapi/*`

As a best practice, avoid adding asynchronous actions in your custom Processing Rules because they increase the use of memory per transaction.



---

## Gateway extension guidelines - DataPower API Gateway

Before you develop your extensions to the DataPower® API Gateway, consider these guidelines.

Note: These guidelines apply to the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), see [Gateway extension guidelines - DataPower Gateway \(v5 compatible\)](#).

For information on the different types of gateway, see [API Connect gateway types](#).

---

### Introduction

You apply an extension to a DataPower API Gateway by uploading one or more `.cfg` files that define the required extension behavior. These `.cfg` files contain DataPower API Gateway CLI commands. For details of the full set of CLI commands that are available, see the [DataPower API Gateway CLI command documentation](#).

You package the `.cfg` files in a `.zip` file, together with any additional required files that are referenced from CLI commands; for example, a `.json` file that contains the OpenAPI definition for an API. The `.cfg` files are processed in alphanumeric order by file name.

Note:

- All the `.cfg` files must be at the root of the `.zip` file. If you are attempting to use an API Connect Version 2018 gateway extension that has files in sub-directories, see [Gateway extensions manifest](#) for information on how to deploy multiple extensions in a single package.
- On Mac OS X, run the `zip` command on the command line with the following flags:

```
-x ".DS_Store" -x "__MACOSX"
```

Including the flags prevents the auto-generated `.DS_Store` and `__MACOSX` files from being added to the zip.

You upload the extension `.zip` file to an API Connect Gateway service, and enable the extension, as described in [Configuring your Gateway server extensions](#). Gateway server extensions can be deployed without having to restart the DataPower API Gateway.

To control how gateway extensions are applied to the DataPower API Gateway, you can include a Gateway extensions manifest as a `manifest.json` file. The manifest allows you to deploy multiple extensions of different types in a single package. For more information, see [Gateway extensions manifest](#).

---

### Using CLI commands in a Gateway extension

Although you can use any DataPower API Gateway CLI commands in your extension `.cfg` files, typically you use commands that modify API Connect objects that have previously been deployed to the DataPower API Gateway.

When API Connect objects are deployed to a DataPower API Gateway, `.cfg` are created on the Gateway that define the configuration of those objects. By examining those `.cfg` on the Gateway, you can determine the precise names of API Connect objects that you want to modify, and the current values of the properties of those objects. You can access these `.cfg` files by completing the following steps:

1. Log in to the DataPower administrative user interface; for the Graphical Interface, select WebGUI rather than Blueprint Console.
2. Switch to the DataPower application domain for API Connect if necessary.
3. Under Files and Administration, click File Management.
4. Expand the temporary: folder to locate the `.cfg` files. The files for API Connect objects are named according to their containing provider organization and Catalog, as follows:

```
20.provider-org-name_catalog-name_collection.cfg
```

The prefix `20` controls the position of the file in the overall processing sequence, which is in alphanumeric order by file name.

Restriction:

- Do not modify these files directly in the DataPower API Gateway file system. Instead, use the gateway extension mechanism described on this page.
- Do not apply gateway extension modifications to configuration in the `local:/config-sequence.cfg` file.
- Do not use Gateway server extensions to modify objects that cause the `apic-gw-service` object to restart, such as TLS Profiles, and Gateway Peering. Extensions that cause the `apic-gw-service` object to restart, lead to unpredictable behavior in the Gateway service.

Note: Modifying a `.cfg` file triggers the execution of all other `.cfg` files that run commands on the same object.

Config-Sequence tracks all of the objects that it manages through the `.cfg` files. Any change to a `.cfg` file that causes config-sequence to run commands on an object automatically executes all other `.cfg` files that run commands on the same object.

Example: If a new product is published to catalog A, then the `.cfg` file for that collection in `temporary://config` will be modified by the `apic-gw-service` to reflect the new product. Because that `.cfg` file was changed, config-sequence will execute every other `.cfg` file that also runs operations on catalog A, in the sequence of locations defined in the config-sequence object. If there are 2 `.cfg` files in the same location that run commands on catalog A, they run in alphabetical order.

## Example 1 - change the scope of the rate limit for a Plan

Suppose the DataPower API Gateway configuration for a Plan has the following CLI command:

```
api-plan myorg_sandbox_financial-services_1.0.0_basic
```

The **api-plan** command has the following syntax:

```
api-plan plan_name
```

and creates a Plan of the specified name, or modifies a Plan of the same name if it already exists. By using the specified name, you can add **api-plan** commands in your Gateway extension .cfg files to modify the Plan.

Note: The names of API Connect objects in the DataPower API Gateway configuration are derived from their configuration in API Connect. For example, a Plan name has the following structure:

```
provider-org-name_catalog-name_product-name_product-version_plan-name
```

where:

- *provider-org-name* is the name of the provider organization.
- *catalog-name* is the name of the Catalog in that provider organization.
- *product-name* is the name of the Product that contains the Plan.
- *product-version* is the version of the Product.
- *plan-name* is the name of the Plan.

The default scope to which a Plan rate limit applies is per application. To change this setting on the Plan in this example, so that the rate limit scope is per client ID, add the following command to a .cfg in your Gateway extension:

```
api-plan myorg_sandbox_financial-services_1.0.0_basic
rate-limit-scope per-client-id
exit
```

For more information on configuring Plans by using DataPower API Gateway CLI commands, see [API Plan commands](#).

## Example 2 - add a path to an API definition

The .cfg file in this example performs the following actions:

1. Uses the **top** and **configure terminal** commands to reset CLI processing.
2. Uses the **api-operation** command as follows:
  - a. Creates a new operation called **bank\_branches\_get\_operation**.
  - b. Uses the **reset** command to set all properties to their default values.
  - c. Sets the operation method to **GET**.
3. Uses the **api-path** command as follows:
  - a. Creates a new path called **bank\_branches\_path**.
  - b. Uses the **reset** command to set all properties to their default values.
  - c. Sets the path URL segment to **/details**.
  - d. Adds the previously created operation **bank\_branches\_get\_operation** to the path.
4. Uses the **api-definition** command as follows:
  - a. Modifies the existing API definition called **myorg\_sandbox\_myapi\_1.0.0**.

Note: The name of the API definition was derived from its configuration in API Connect when it was published to the DataPower API Gateway, and has the following structure:

```
provider-org-name_catalog-name_api-name_api-version
```

where:

- *provider-org-name* is the name of the provider organization.
- *catalog-name* is the name of the Catalog in that provider organization.
- *api-name* is the name of the API definition in API Connect.
- *api-version* is the version of the API.

- b. Adds the previously created path **bank\_branches\_path** to the API definition.

Note: The **reset** command is not used here because that would reset all the current settings on the **myorg\_sandbox\_myapi\_1.0.0** API definition, whereas the requirement is to add a new path to the existing configuration of the API definition.

The .cfg file is as follows:

```
top; configure terminal

api-operation bank_branches_get_operation
reset
method GET
exit

api-path bank_branches_path
reset
path "/details"
operation bank_branches_get_operation
exit

api-definition myorg_sandbox_myapi_1.0.0
path bank_branches_path
exit
```

For more information on the DataPower API Gateway CLI commands used in this example, see the following pages:

- [Initial login and common commands](#)
- [API Operation commands](#)

- [API Path commands](#)
- [API Definition commands](#)

## Gateway extension limitations

This section details the limitations of the extensions to the DataPower API Gateway.

- If you use an extension to create a user-defined policy and then delete the extension, the extension is removed from the filesystem, but the configuration sequence is not triggered to clean up related files. For the policy to be cleaned up, you must send a modified extension that explicitly removes the user-defined policy from `apic-gw-service`, as in the following example.

```
apic-gw-service
no user-defined-policies udp-basic_1.0.0
exit
```

- `user-defined-policy-yaml` policies that use or reference another policy can be deployed only after the referenced policy has been advertised to API Manager. The deploy type must be set to `deferred` and the API Connect Gateway Service must be restarted.
- Deleting an extension that is used to create a user-defined policy using a `.cfg` file does not remove the policy reference in the `apic-gw-service` object. The policy is not removed from the assembly palette until a reboot or restart of the Gateway pod. To properly remove the policy, send an `apic extensions:update` command that explicitly removes the policy from the `apic-gw-service` prior to sending the `apic extensions:delete` command.
- v5 policies that contain files outside of the `/policy` directory are not supported. If multiple v5 policies reference the same file outside the `/policy` directory, the last applied policy will overwrite the file and will be used by all policies that reference it. Removing any policy that contains the file will cause the file to be deleted and may cause errors for the remaining policies that use it.
- If an extension contains or references objects and files that are also referenced by other objects outside of the extension, deleting the extension can cause errors that are difficult to troubleshoot.
- It is possible to delete an extension that is referenced by published APIs. This may cause those API requests to return errors.
- For an extension type of `user-defined-policy`, extra assembly functions may be present in the DataPower import `.zip` file, even if there is no `.yaml` file to indicate their presence. Using these functions without declaring them using a `.yaml` file is not supported. A policy `.yaml` file is required for each policy you want to advertise to API Manager.
- Extension type `user-defined-policy-yaml` may leak generated Gatewayscript `.js` files in temporary `/js/` or `.xslt` files when deleted.
- Extension types that use SOMA imports must use new objects with no overrides. If you override a singleton or shared object using an extension of this type, then the object will not be restored to its previous state when the extension is deleted, and those objects may be missing after the extension is deleted. You must then restore any needed objects.
- [Gateway extensions manifest](#)  
A Gateway extensions manifest allows you to control how gateway extensions are applied to the DataPower API Gateway.
- [v5 policy emulation limitations](#)  
This page details the limitations of v5 policy emulation for the current release.



## Gateway extensions manifest

A Gateway extensions manifest allows you to control how gateway extensions are applied to the DataPower® API Gateway.

The manifest allows you to deploy multiple extensions of different types in a single package. It is a JSON file that lists all of the extensions `.zip` files that are to be applied and instructions on when to deploy them. You can also use the manifest to enable v5 policy emulation for the API Connect Gateway Service. v5 policy emulation allows you to deploy supported v5 built-in policies and custom policies, and manage them from API Manager without having to rewrite them for DataPower API Gateway. v5 policy emulation is supported only for policies that are applied to the Gateway service scope.

The `manifest.json` file must be at the root level of the extensions `.zip` file with all of the Gateway extension files specified in the manifest.

The `manifest.json` file contains the following sections.

### properties section

The properties section determines whether to enable v5 framework to deploy supported v5 built-in policies and custom policies and manage them from API Manager without having to rewrite them for DataPower API Gateway.

Table 1. `properties` section properties

Property	Description
<code>deploy-policy-emulator</code>	Determines whether to enable v5 framework emulation. Valid values are <code>true</code> and <code>false</code> .  v5 framework emulation is enabled for the API Connect Gateway service if <code>deploy-policy-emulator</code> set to <code>true</code> , if a v5 policy is detected in the files list, or if a <code>deploy-policies</code> entry is specified.
<code>deploy-policies</code>	Contains a list of built-in policies to advertise for v5 emulation. The following policies are supported. <ul style="list-style-type: none"> <li>• <code>activity-log_1.5.0</code></li> <li>• <code>gatewayscript_1.0.0</code></li> <li>• <code>if_1.5.0</code></li> <li>• <code>invoke_1.5.0</code></li> <li>• <code>proxy_1.5.0</code></li> <li>• <code>redact_1.5.0</code></li> <li>• <code>switch_1.5.0</code></li> <li>• <code>validate-usenametoken_1.0.0</code></li> <li>• <code>xslt_1.0.0</code></li> </ul>

## files section

The `files` section is an array of file entries. Each file entry must reference a file that is in the `extension.zip` file.

Table 2. `files` section properties

Property	Description
<code>filename</code>	The location of the file in the <code>.zip</code> file. This property is required only if files are specified.
<code>type</code>	<p>The file type. This property is required.</p> <p>The following types are supported.</p> <p><b>dp-import</b> A gateway import used to add other services to a gateway. Note: A <code>dp-import</code> file cannot be used to change the <code>apic-gw-service</code> object. If it is used for this purpose, the extension is rejected.</p> <p><b>extension</b> A complete, valid gateway extension for an API Gateway. The extension must be a <code>.zip</code> file that contains <code>.cfg</code> files. For more information, see <a href="#">Gateway extension guidelines - DataPower API Gateway</a>.</p> <p><b>policy-v5</b> A complete, valid v5 policy implementation. For more information, see <a href="#">Authoring policies for the DataPower Gateway (v5 compatible)</a>.</p> <p><b>user-defined-policy</b> A valid API Gateway policy import <code>.zip</code> file that contains one or more policies. The <code>.zip</code> file contains one or more YAML policy definition files. The YAML files are used to indicate which policies to advertise to the API Manager server as service-level user policies. If a YAML file references an assembly function that does not exist in the implementation, the extension is rejected. The <code>.zip</code> file also contains an implementation directory that includes a policy import <code>.zip</code> file. The policy import file contains all the configurations and files necessary to run the user policies, including the assembly functions to advertise to the API Manager.</p> <p><b>user-defined-policy-yaml</b> A YAML file that includes an assembly. This assembly is converted to an assembly function and advertised to the API Manager at the gateway service user policy level. If the assembly references another user policy, that policy must already be advertised to the API Manager before it can be deployed.</p>
<code>deploy</code>	<p>When to deploy the file.</p> <p><b>immediate</b> Deploy the file as soon as it is received by the API Connect Gateway Service.</p> <p><b>deferred</b> Deploy the file after the API Connect Gateway Service is restarted. This value is the default setting.</p>
<code>catalog</code>	Indicates that a v5 policy is to be used only by certain catalogs and should not be advertised as a gateway service-level user policy. This property is valid only if the file type is <code>policy-v5</code> . The <code>catalog</code> value must be in the format <code>orgname_catalogname</code> .

## Example `manifest.json`

The following example `manifest.json` file enables v5 policy emulation and specifies two built-in v5 policies to deploy. The `files` section lists two v5 policy implementation files and two extension files.

```
{
 "extension": {
 "properties": {
 "deploy-policy-emulator": true,
 "deploy-policies": ["invoke_1.5.0", "proxy_1.5.0"]
 },
 "files": [
 {
 "filename": "jws-sign-policy.zip",
 "deploy": "deferred",
 "type": "policy-v5",
 "catalog": "testorg_testcatalog"
 }, {
 "filename": "jws-verify-policy.zip",
 "deploy": "immediate",
 "type": "policy-v5"
 }, {
 "filename": "APIGWLegacyExt_1.zip",
 "deploy": "deferred",
 "type": "extension"
 }, {
 "filename": "APIGWLegacyExt_2.zip",
 "deploy": "immediate",
 "type": "extension"
 }
]
 }
}
```



## v5 policy emulation limitations

This page details the limitations of v5 policy emulation for the current release.

- The reverse solidus character (`\`) may be removed from policy property values.

- The source parameter for XSLT and GatewayScript policies is base-64 encoded to preserve escaping. It cannot be easily viewed from the DataPower command line or the DataPower WebGUI.
- Some DataPower service variables (for example, `var://service/mpgw/skip-backside`) may not be applicable in API Gateway, depending on how the variables are used.
- User-defined policies that make any alterations to a DataPower context are not supported.
- Execution of a 1.0.0 policy may fail if a previous 2.0.0 policy updates the `message.body` content type (for example, from XML to JSON, or JSON to XML).
- A proxy 1.5.0 policy can return an incorrect response if it is the last action within a `switch case` or `otherwise` property. The response is parsed. The correct behavior is to not parse the response. To work around this issue, set the `final-action: true` parameter on the proxy.
- Proxy 1.5.0 and invoke 1.5.0 policies do not preserve the `gtid` header.
- The `cache-putpost-response` property is not included in the schema for invoke 1.5.0 and proxy 1.5.0 policies.
- Setting HTTP response headers in a v5 user-defined policy, GatewayScript policy, or XSLT policy is not supported.
- Non-UTF-8 characters are not supported.
- JSON data returned in the response of an API running in v10 with v5 emulation is formatted without spaces.

## Configuring your Gateway server extensions



You can add extra IBM® DataPower® enforcement capabilities to a Gateway service by uploading a .zip file that defines the required extension behavior, and then enabling the extension in IBM DataPower.


### About this task

You can upload a Gateway server extension to a Gateway service in either of the following ways:

- [Use the Cloud Manager user interface](#)
- [Use the developer toolkit CLI](#)

### Procedure

- To upload a Gateway server extension to a Gateway service by using the Cloud Manager user interface, complete the following steps:
  1. In the Cloud Manager user interface, click  Topology.
  2. Alongside the required Gateway service, click the options icon , then click Configure gateway extension.
  3. Click Add.
 

Note: You cannot upload more than one Gateway extension .zip file to the same Gateway service. If a Gateway server extension has previously been uploaded, the Add button is not available; you can replace the previous extension by clicking the options icon  alongside gateway-extension, then clicking Edit.
  4. Upload the extension as indicated, either by dragging and dropping the .zip file, or by browsing and selecting it.
  5. Click Save when done.
- To upload a Gateway server extension by using the developer toolkit CLI, complete the following steps:
  1. Log in to your API Connect Cloud Manager server as an administrator, using the following command:

```
apic login --server <cloud-manager_endpoint_url> --username <admin_user_ID> --password <admin_password> --realm admin/<identity_provider>
```

where:

- `<cloud-manager_endpoint_url>` is the Cloud Manager endpoint URL; for example: `mgmt-admin.<myserver>.com`.
- `<admin_user_ID>` is the user ID of your administrator account, and is the same as the user ID that you use to log in to the Cloud Manager user interface.
- `<admin_password>` is the password for your administrator account.
- `<identity_provider>` is the identity provider that is used to authenticate administrative users.

For example:

```
apic login --server mgmt-admin.myserver.com --username admin --password my-admin-password --realm admin/myldap
```

2. Upload the Gateway extension .zip file by using the following command:

```
apic gateway-extensions:create <extension_zip_file> --scope org --org admin --gateway-service <gateway_service> --availability-zone <availability-zone> --server <cloud-manager_endpoint_url>
```

where:

- `<extension_zip_file>` is the Gateway extension .zip file that you want to upload.
- `<gateway_service>` is the name of the Gateway service that you want add the extension to.
- `<availability-zone>` is the name of the availability zone that contains the Gateway service.
- `<cloud-manager_endpoint_url>` is the Cloud Manager endpoint URL.

For example:

```
apic gateway-extensions:create myextension.zip --scope org --org admin --gateway-service mygatewayservice --availability-zone availability-zone-default --server mgt-admin.myserver.com
```

This example uses the default supplied availability zone name of `availability-zone-default`, which will be the required value if you have not configured your own availability zones. To check the names of the currently configured availability zones, use the following command:

```
apic availability-zones:list --org admin --server <cloud-manager_endpoint_url>
```

For details on configuring availability zones, see [Creating an availability zone](#).

To check the names of the currently configured gateway services, use the following command:

```
apic gateway-services:list --org admin --availability-zone <availability-zone> --server <cloud-manager_endpoint_url>
```

You can confirm that the extension has been added to the Gateway service by using the `gateway-extensions:get` command; for example:

```
apic gateway-extensions:get --scope org --org admin --gateway-service mygatewayservice --availability-zone
availability-zone-default --server mgt-admin.myserver.com --output -
```

(the parameter setting `--output -` writes the details of the Gateway extension object to the command window. You can specify the name of an existing folder to have the details written to a `.yaml` file in that folder.)

For reference details of the `apic gateway-extensions` commands, see [apic gateway-extensions](#).

Note: You cannot upload more than one Gateway extension .zip file to the same Gateway service. If you want add further extensions later, update the original .zip file, then use the `apic`

`gateway-extensions:update` command to replace the previous gateway extensions file with the revised one; for example:

```
apic gateway-extensions:update mynewextension.zip --scope org --org admin --gateway-service mygatewayservice --
availability-zone myavailabilityzone --server mgt-admin.myserver.com
```

3. Apply the extension to each server by restarting the API Connect gateway service object; complete the following steps on **each** Gateway server in the Gateway service:

Note: In Kubernetes environments, the following steps can be accomplished by issuing `kubectl`

`delete pod` on **each** gateway pod. Be sure to wait until each pod gets back to the ready state before deleting the next gateway pod.

- a. Remove the Gateway server from the load balancing group.
- b. Disable, then enable the API Connect gateway service object. You can do this by using either the DataPower administrative user interface in a web browser, or by using the DataPower CLI.

- By using the administrative user interface:

- Log in to the DataPower administrative user interface; for the Graphical Interface, select WebGUI rather than Blueprint Console.
- Switch to the DataPower application domain for API Connect if necessary.
- Search for API Connect Gateway Service.
- Set the Administrative State to disabled.
- Apply the changes.
- Set the Administrative State to enabled.
- Apply the changes.

- By using the CLI:

- Enter configuration mode by entering the command `configure`.
- Navigate to the DataPower application domain for API Connect by entering the command `switch api_connect_domain_name`, where `api_connect_domain_name` is the name of your API Connect application domain.
- Disable the API Connect gateway service object by entering the following command:

```
apic-gw-service; admin-state disabled; exit
```

- Enable the API Connect gateway service object by entering the following command:

```
apic-gw-service; admin-state enabled; exit
```

- c. Ensure that the gateway service object initialization has completed.

- d. Re-add the Gateway server to the load balancing group.

4. Apply the extension to each server by restarting the API Connect gateway service object; complete the following steps on **each** Gateway server in the Gateway service:

Note: In Kubernetes environments, the following steps can be accomplished by issuing `kubectl`

`delete pod` on **each** gateway pod. Be sure to wait until each pod gets back to the ready state before deleting the next gateway pod.

## Results

The extension is uploaded and applied to each of the servers in the Gateway service, and the associated enforcement capabilities are applied to all incoming API resource requests.

Warning: **DataPower Gateway (v5 compatible) only** If you upload an extension to the DataPower Gateway (v5 compatible), a status file called `extension_import_response.xml` is written to the `local:/ext` folder in the gateway file system. Do not remove this file, otherwise if your gateway extension is subsequently removed then the referenced objects cannot be reverted to their original state automatically, and you will therefore have to either complete these clean up tasks manually, or re-add the gateway extension to regenerate the `extension_import_response.xml` file and then remove the gateway extension again. If the `extension_import_response.xml` exists and there is a cleanup failure for any object, a failure file is written to the `local:/ext` folder, with the failure details.

## Cloud Manager Tutorials

Tutorials for using the Cloud Manager user interface in IBM® API Connect.

## Prerequisites

To complete the following tutorials you must be the Cloud Administrator or Cloud Owner of an API Connect instance.

## Overview

These tutorials guide you through common tasks performed to set up and maintain the configuration of a cloud.

- [Tutorial: Configuring the Cloud](#)  
This tutorial shows you how to create a basic cloud configuration, with available gateway, analytics and developer portal services.
- [Tutorial: Creating a Provider Organization](#)  
This tutorial shows you how to create a Provider Organization.

---

## Tutorial: Configuring the Cloud

This tutorial shows you how to create a basic cloud configuration, with available gateway, analytics and developer portal services.

### Before You Begin

---

This task can be completed by users who are assigned one of the following roles:

- Cloud Owner
- Cloud Administrator

You will need the following information to complete this tutorial.

- The IP address or FQDN of the gateway, and the port assigned to accept API requests from clients.
- The IP address or FQDN of the gateway, and the port assigned to communicate with the API Management server. This cannot be the same as the port for API requests.
- The FQDN of the Developer Portal service to accept requests from clients.
- The FQDN of the Developer Portal to communicate with the API Management server.
- The FQDN of the Analytics service to accept requests from clients.
- The address, port and optional login credentials for an SMTP email server.

Note: The port for the `portal-ww`, `portal-admin`, and `analytics-ingestion` services is **443**, and isn't configurable.

### About this tutorial

---

In this tutorial you are going to complete the following lessons:

- [Initial Cloud Manager Console login](#)
- [Configure an Email Server](#)
- [Register a Gateway Service](#)
- [Register a Portal Service](#)
- [Register an Analytics Service](#)
- [Configure a Default Gateway Service](#)

### Initial Cloud Manager Console login

---

Take the following steps to log in to the Cloud Manager user interface for the first time.

1. In a web browser, enter the management service URL. For example, `https://ManagementService.domain/admin` where `ManagementService.domain` is the fully qualified host name or IP address of the Management service.

IBM API Connect

## Cloud Manager

Sign in using the Cloud Manager User Registry

Username

Password

Sign In

[Forgot Password?](#)

2. Enter the Cloud Administrator user name and password. The default values are `admin` for the user name and `7iron-hide` for the password.
3. You are immediately required to change the admin password, as well as provide an email address for the cloud administrator. Enter the necessary information.  
Note: If you forget your password and request a password reset, the notification email is sent to this email address. This action will use the email server set in the Notifications section of the cloud Settings. This tutorial shows you how to set this configuration.

## Change Password

Email

Current password

New Password


Confirm password

Save

4. Click Save.
5. Log in using the new password.










## Configure an Email Server

Take the following steps to configure an email server. This configuration allows you to recover the admin password or receive other important notifications.

1. Click  Resources icon or the Manage Resources tile.  
Welcome to Cloud Manager

Let's get you up and running



 Configure Cloud  Edit settings for user registries, roles, endpoints, and more	 Configure Topology  Manage availability zones and services	 Manage Resources  Configure user registries, TLS, OAuth providers and email servers	 Manage Organizations  Create and manage API provider organizations and owners
  Learn More  Documentation and tutorials with step-by-step instructions	  Connect  Find expert answers in the API Connect community forum	 Download Toolkit  Download toolkit and credentials for various platforms	

2. Click Notifications.



## Resources

User Registries Create

<input type="checkbox"/>	Title	Type	Visible To	
<input type="checkbox"/>	<a href="#">API Manager Local User Registry</a>	Local User Registry	Private	:
<input type="checkbox"/>	<a href="#">Cloud Manager Local User Registry</a>	Local User Registry	Private	:

Items per page 10 ▾ 1-2 of 2 items 1 ▾ 1 of 1 pages ◀ ▶


3. Click Create.

## Resources

User Registries  
TLS  
OAuth Providers  
**Notifications**

Email Servers Create

<input type="checkbox"/>	Title	Mail Server
--------------------------	-------	-------------

 **You currently don't have any Email Servers.**  
Click Create to add a new Email Server.

4. Enter the appropriate values in the fields These values vary depending on your site.

5. Click Save.

## Create Email Server

### Email Server Configuration

Title  
My Email Server

Name  
my-email-server

Address  
myemailserver.net

Port  
587

Authenticate User (optional)

Authenticate Password (optional)




TLS Client Profile (optional)  
Default TLS client profile

Secure Connection

## Test Connection

Send a test email to confirm that the email server is properly configured.

Test email

6. Click the  Settings icon, then click Notifications  Edit.
7. Ensure your email server is selected, then click Save.
8. Click the  Home icon.

## Register a Gateway Service

---








Take the following steps to register a gateway service.

1. Click the Configure Topology tile.

Welcome to Cloud Manager

Let's get you up and running



 Configure Cloud  Edit settings for user registries, roles, endpoints, and more	 Configure Topology  Manage availability zones and services	 Manage Resources  Configure user registries, TLS, OAuth providers and email servers	 Manage Organizations  Create and manage API provider organizations and owners
 Learn More  Documentation and tutorials with step-by-step instructions	 Connect  Find expert answers in the API Connect community forum	 Download Toolkit  Download toolkit and credentials for various platforms	

2. Click Register Service.


## Topology

Configure availability zones and services

Create Availability Zone

Default Availability Zone Management

Register new services and manage existing services Register Service

Service	Type	Associated Analytics Service	Visible To
 <b>You currently don't have any Services.</b> Click Register Service to register a service. This will allow you to test endpoints when configuring other services. <a href="#">Learn more</a>			

3. Click DataPower API Gateway.

## Configure Service

Select Service Type

<b>DataPower API Gateway</b>  Configure a DataPower API gateway service for securing and enforcing APIs	<b>DataPower Gateway (v5 compatible)</b>  Configure a DataPower gateway service for securing and enforcing APIs	<b>Portal</b>  Configure a developer portal service for API consumers	<b>Analytics</b>  Configure an analytics service to collect API call data
---------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------	---------------------------------------------------------------------------------

Cancel

4. Take the following steps.

- Enter `gateway_service` in the Title field.
- In the Management Endpoint section, enter the URL of the address and port assigned to the management endpoint in the Endpoint field. This is the port used by the API Management server to connect to the gateway.
- Leave the remaining values as given to set TLS profiles.

## Configure DataPower API Gateway Service

**Gateway Details**

Title  
gateway\_service

Name  
gateway-service

Summary (optional)

**Management Endpoint**

Endpoint  
https://gateway.example.com:3000

TLS Client Profile  
Default TLS client profile

- d. In the API Invocation Endpoint section, enter the URL of the gateway address and port assigned to accept API requests from clients in the API Endpoint Base field.
- e. Do not change the defaults in the Server Name Indication (SNI) fields. Note that your topology may require specific values in these fields.
- f. Optionally enter a hex value in the OAuth Shared Secret field. This must be a 64-bit hex value that begins with "0x". Providing a shared secret value here enables gateways in a cluster to read the OAuth tokens generated by any member of the cluster using the same secret.
- g. Click Save.

**API Invocation Endpoint**

API Endpoint Base  
https://apigw.example.com:443

Server Name Indication (SNI) Add

Host name	TLS Server Profile	Order	Delete
*	Default TLS server profile		

OAuth Shared Secret (optional)  
0x

## Register a Portal Service

Take the following steps to register a portal service.

1. Click Register Service.
2. Click Portal.
3. Take the following steps.
  - a. Enter `portal_service` in the Title field.
  - b. In the Management Endpoint section, enter the URL of the portal address and port assigned to communicate with the management server in the Endpoint field.
  - c. Enter the URL of the portal address and port assigned to accept requests from clients in the Portal Website URL field.
  - d. Use the reconfigured profile in the TLS Client Profile field.
  - e. Click Save.

## Configure Portal Service

### Portal Details

---

Title  
portal\_service

Name  
portal-service

Summary (optional)

---

### Management Endpoint

---

Endpoint  
https://portalsvc.example.com:9443

TLS Client Profile  
Portal Director TLS client profile

Portal Website URL  
https://portalsvc.example.com:443

## Register an Analytics Service

---

Take the following steps to register an analytics service.

1. Click Register Service
2. Click Analytics.
3. Take the following steps.
  - a. Enter `analytics_service` in the Title field.
  - b. In the Management Endpoint section, enter the URL of the analytics server address and port assigned to accept requests from clients in the Endpoint field.
  - c. Select `Analytics ingestion TLS client profile` in the TLS Client Profile field.
  - d. Click Save.

## Configure analytics service

### Analytics details

Title  
analytics\_service

Name  
analytics\_service

Summary (optional)

---

### Management endpoint on the analytics service

Endpoint  
Enter the fully-qualified domain name for the Analytics ingestion endpoint that you defined during installation.  
https://example.analytics.com

TLS client profile  
Analytics ingestion TLS client profile:1.0.0

[Cancel](#) [Save](#)

- Click Associate Analytics Service corresponding to the gateway service listing.

Default Availability Zone Management [Register Service](#)

Register new services and manage existing services

Service	Type	Associated Analytics Service	Visible To
portal_service	Portal Service		Public
gateway_service	DataPower API Gateway	<a href="#">Associate Analytics Service</a>	Public
analytics_service	Analytics Service		

- Select analytics\_service. Click Associate.

## Associate Analytics Service

### Gateway Service

gateway\_service

### Analytics Service

Select the analytics service you would like to associate with the gateway service

ANALYTICS	AVAILABILITY ZONE
<input checked="" type="checkbox"/> analytics_service	availability-zone-default

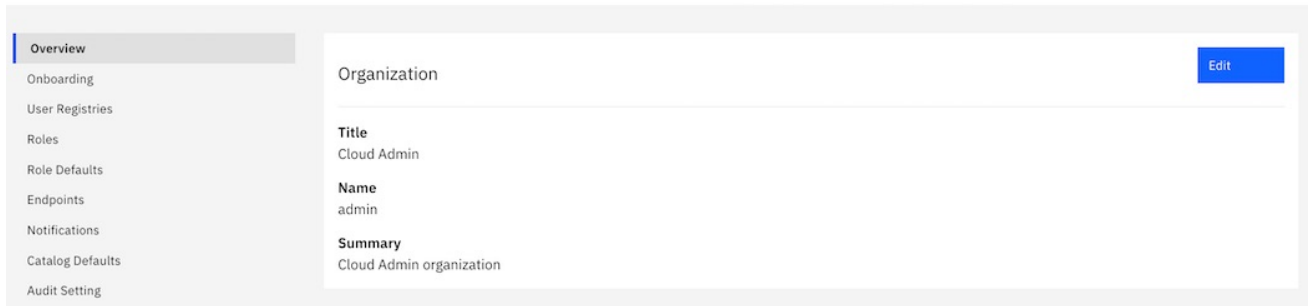
[Cancel](#) [Associate](#)

## Configure a Default Gateway Service

Take the following steps to configure a default gateway service for every catalog in the cloud.

1. Click  Settings icon .
2. Click Catalog Defaults.

## Settings

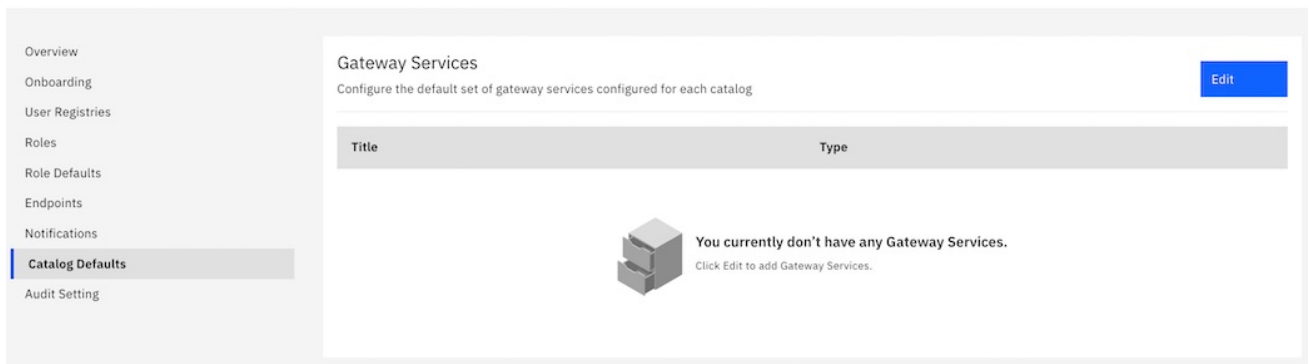


The screenshot shows the 'Settings' page with a sidebar menu on the left containing: Overview, Onboarding, User Registries, Roles, Role Defaults, Endpoints, Notifications, Catalog Defaults, and Audit Setting. The 'Catalog Defaults' menu item is selected. The main content area is titled 'Organization' and includes an 'Edit' button in the top right corner. Below the title, the following information is displayed:

- Title:** Cloud Admin
- Name:** admin
- Summary:** Cloud Admin organization


3. Click Edit.

## Settings



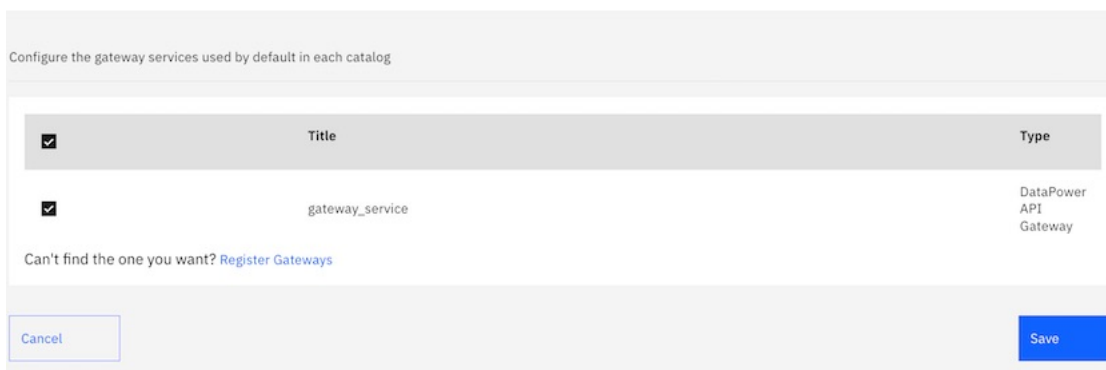
The screenshot shows the 'Settings' page with the 'Catalog Defaults' menu item selected. The main content area is titled 'Gateway Services' and includes an 'Edit' button in the top right corner. Below the title, the following information is displayed:

Configure the default set of gateway services configured for each catalog

Title	Type
 <p>You currently don't have any Gateway Services. Click Edit to add Gateway Services.</p>	

4. Select an available gateway service.
5. Click Save.

## Edit Gateways



The screenshot shows the 'Edit Gateways' dialog box with the following content:

Configure the gateway services used by default in each catalog

<input checked="" type="checkbox"/>	Title	Type
<input checked="" type="checkbox"/>	gateway_service	DataPower API Gateway

Can't find the one you want? [Register Gateways](#)

Buttons: Cancel, Save

## What you did in this tutorial

- Set new Cloud Manager Console login password
- Configured a Gateway Service
- Configured a Developer Portal Service
- Configured an Analytics Service
- Configured a Notifications Email Server
- Configured a Catalog Default Gateway Service

## Related information

- [All tutorials](#)

## Tutorial: Creating a Provider Organization

This tutorial shows you how to create a Provider Organization.

## Before You Begin

This task can be completed by users who are assigned one of the following roles:

- Cloud Owner
- Cloud Administrator

You must also complete the following tasks before beginning:

- [Configuring an email server for notifications](#)
- [Setting up notifications](#)

Note that an email server must be configured and an active email server must be selected before a provider organization account can be created.


## About this tutorial

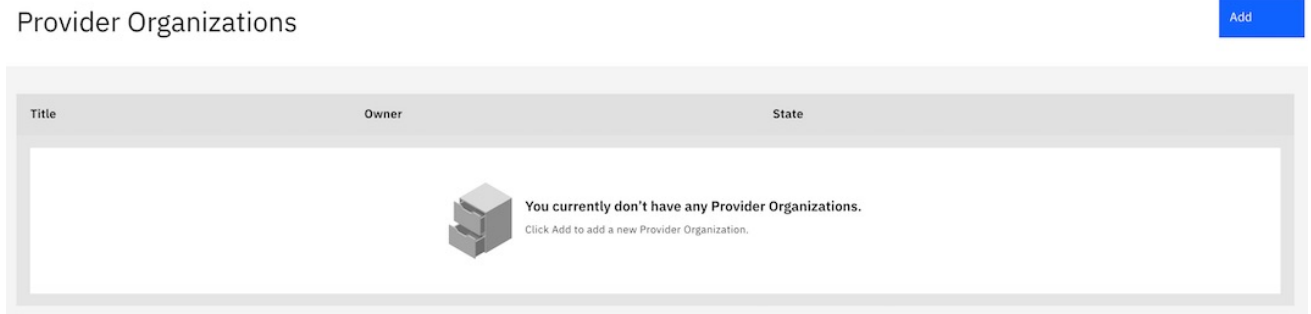
In this tutorial you are going to complete the following lessons:

- [Create a Provider Organization](#)

## Create a Provider Organization

Take the following steps to create a new Provider Organization.

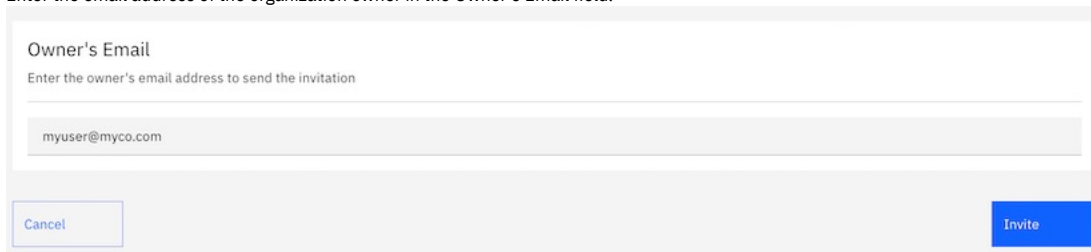
1. Log in to the Cloud Manager.
2. Click  Provider Organizations.



3. Click Add > Invite organization owner.

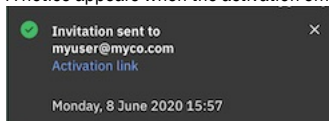



4. Enter the email address of the organization owner in the Owner's Email field.

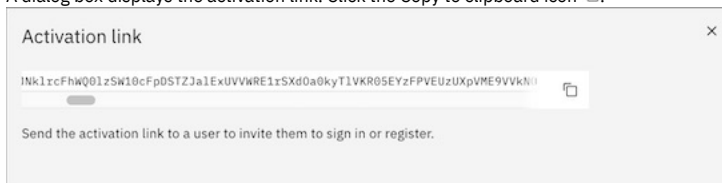


The screenshot shows the 'Owner's Email' form. The text 'Enter the owner's email address to send the invitation' is above the input field. The email address 'myuser@myco.com' is entered in the field. There are 'Cancel' and 'Invite' buttons at the bottom.

5. Click Invite.
6. A notice appears when the activation email has been sent. Click Activation link..



7. A dialog box displays the activation link. Click the Copy to clipboard icon .





- Open a new window in your browser. Paste the copied activation link into the Location bar. Optionally, you can click the activation link in the invitation email to open the registration form.
- Complete fields shown in the form to sign up with the API Manager User Registry. Click Sign up.

IBM API Connect

## API Manager

Sign up with API Manager User Registry

New Organization Title

Tutorial

Username

tutor

Email

myuser@myco.com

First Name

Apic

Last Name

Tutorial

Password

.....

Confirm password

Sign Up

Already have an account? [Sign in](#)

- You see a confirmation of registration. Click Sign in.

IBM API Connect

## API Manager

**Registration completed successfully**

Congratulations, you are now registered.

**Sign in to:**

- Manage users
- Administer consumer organizations
- Create and publish APIs and Products
- Analyze API usage and performance
- Manage API life cycles

Sign In

- Enter the username and password you just created. Click Sign in to begin creating APIs and products.

# IBM API Connect API Manager

Sign in using the API Manager User Registry

Username


tutor

Password

\*\*\*\*\*

Sign In

[Forgot Password?](#)

12. Return to the original window of your browser. Click  Provider Organizations to refresh the list. Your new organization is listed.

## Provider Organizations

Add

Title	Owner	State
Tutorial	 Apic Tutorial myuser@myco.com	Enabled

Items per page 10 1-1 of 1 items 1 1 of 1 pages

## What you did in this tutorial

In this tutorial, you completed the following activities:

- Created a new Provider Organization.

## Related information

- [Importing an API](#)

## Developing your APIs and applications

You develop APIs and LoopBack® applications by using the IBM® API Connect developer toolkit.

The API Connect developer toolkit provides both the API Designer user interface and a command line interface that you can use to develop APIs and LoopBack applications and publish them to API Connect.

You publish APIs by including them in a Product and then publishing the Product. You define your APIs and Products by creating and validating YAML definition files in your local file system. You can then interact with API Connect by using either the API Designer or the toolkit commands.

The developer toolkit is described in detail in the following subsections:

- [API development checklist](#)  
A summary, with links, of the key tasks for developing APIs in IBM API Connect.
- [Working with the toolkit](#)  
Install the developer toolkit by downloading an executable file. You can then use the toolkit commands to interact with IBM API Connect and publish APIs that you have defined in your file system.
- [Managing platform REST API keys](#)  
Create API keys associated with your user account, and then use them to authenticate when you call the platform REST APIs provided by API Connect.
- [Searching for items in API Manager](#)  
Use the search feature in IBM API Connect API Manager to easily locate items such as APIs, Catalogs, Spaces, applications, and subscriptions.
- [Working with API definitions](#)  
An API definition specifies the complete configuration for an API. You can create and configure your APIs either by using either the API Designer UI application, or by using the browser based API Manager UI.
- [Working with Products](#)  
In IBM API Connect, Plans and APIs are grouped together in Products, with which you can manage the availability and visibility of APIs and Plans.
- [Creating and validating API and Product definitions by using the CLI](#)  
The developer toolkit of IBM API Connect provides a command line interface that you can use to create and publish API and Product definitions, and also to validate YAML or JSON definitions.
- [Working with global policies](#)  
Use global policies to configure policy assemblies that are called just before, or just after, each of your API assemblies is called, or when an error is thrown. You can

upload global policies into each of the gateway services in your Catalogs, and then designate, for each gateway service, which global policy should be called before an API assembly is called, after an API assembly is called, or if an error occurs when an API assembly is called. The designated global policies are applied to all the APIs that are deployed to the associated gateway service.

- [Reference](#)

Reference information for developing your APIs in API Connect, including context variables, and template variables.

## API development checklist

A summary, with links, of the key tasks for developing APIs in IBM® API Connect.

Task	Description
<a href="#">Activate your API Manager user account</a>	For you to be able to connect to a Management server by using either the API Designer or API Manager user interfaces, one or other of the following prerequisites must be satisfied: <ul style="list-style-type: none"> <li>• Your administrator has created a provider organization, specifying you as the owner; see <a href="#">Creating a provider organization</a>.</li> <li>• You have been added as a member of a provider organization; see <a href="#">Adding provider organization users and assigning roles</a>.</li> </ul>
<a href="#">Install the developer toolkit</a>	The developer toolkit provides a command-line environment and an API Designer user interface, both of which enable you to develop and work with your APIs on your local machine; you publish your APIs to the Management server when ready. You can also develop your APIs directly on the Management server by using the API Manager user interface.
Log in to the user interface; see <a href="#">Logging in from API Designer</a> , or <a href="#">Accessing the API Manager user interface</a> , depending on which user interface you want to use.	After you have activated your account, you can log in to your Management server from either user interface. You can also use the API Designer in offline mode, but to be able to test and publish APIs you must be connected to a Management server.
<a href="#">Create your API definitions</a>	You can create API definitions in the following ways: <ul style="list-style-type: none"> <li>• Create a REST API definition either by composing the API definition, and its operations, from scratch, creating a proxy API that calls an existing endpoint, or importing an OpenAPI definition.</li> <li>• Expose an existing SOAP service as an API.</li> <li>• Create a GraphQL API proxy definition that proxies a backend GraphQL server.</li> </ul>
<ul style="list-style-type: none"> <li>• <a href="#">Secure your APIs (OpenAPI 2.0)</a></li> <li>• <a href="#">Secure your APIs (OpenAPI 3.0)</a></li> </ul>	You configure security for an API by creating one or more security definitions that you then apply to your API, and to the operations in your API. Security mechanisms include basic authentication through a user registry, and token based authentication with OAuth.
<a href="#">Test your APIs</a>	The user interfaces provide a test tool with which you can test your APIs through the gateway.
<a href="#">Productize and publish your APIs</a>	To make your APIs available to application developers, you include them in a Product that you then publish to the gateway.
<a href="#">Apply visibility controls and rate limiting to your APIs</a>	Visibility settings control which APIs the application developers can subscribe to. Rate limiting controls the usage of your APIs; for example, you can apply a rate limit of 100 calls per minute.
<a href="#">Develop your APIs by using the developer toolkit CLI</a>	You can use the developer toolkit CLI to develop APIs based on <a href="#">OpenAPI definition files</a> in YAML or JSON format.

## Working with the toolkit

Install the developer toolkit by downloading an executable file. You can then use the toolkit commands to interact with IBM® API Connect and publish APIs that you have defined in your file system.

The following subsections describe how to install and use the toolkit:

- [Installing the toolkit](#)

You can install the toolkit that provides CLI commands, and the API Designer user interface, for IBM API Connect.

- [Logging in from API Designer](#)

To log in from the API Designer user interface for IBM API Connect, you will need the host name of the management server and the user name and password associated with your API Manager account.

- [Using the developer toolkit command-line tool](#)

The IBM API Connect developer toolkit provides a command-line tool, `apic`, that you can use to perform all API Connect tasks. You can also use the command-line tool to script tasks such as continuous integration and delivery.

## Installing the toolkit

You can install the toolkit that provides CLI commands, and the API Designer user interface, for IBM® API Connect.

### About this task

The toolkit is provided as executable files, so no actual installation is necessary, you just need to download the required compressed file and extract the contents.

There are two toolkit options available:

- CLI: provides a command line environment for working with IBM API Connect.
- CLI + LoopBack + Designer: provides a command line environment for working with IBM API Connect, including LoopBack® support, and the API Designer application.

To install the toolkit, download the compressed file that is appropriate for your chosen toolkit option and platform, then extract the contents to a chosen location on your local machine. The compressed file contains an executable file for running CLI commands and, if you choose the CLI + LoopBack + Designer option, an executable file for launching the API Designer application.

You can download the toolkit from one of the following locations:

- From IBM Fix Central.
- From the Cloud Manager or API Manager user interface.

## Procedure

To install and run the toolkit, complete the following steps:

### 1. Download the toolkit compressed file.

- To download the toolkit from IBM Fix Central, complete the following steps:
  - Open the [IBM Fix Central site](#) in your browser.
  - In the Product selector field, enter `API Connect`, then select IBM API Connect from the drop down list.
  - Select your Installed Version and click Continue. If you do not know your installed IBM API Connect version, contact your administrator.
  - In the Text field, enter `toolkit`, then click Continue.
  - Select the toolkit file that you want to download.
  - Click Continue, then follow the instructions to complete the download operation.
- To download the toolkit from Cloud Manager or API Manager user interface, complete the following steps:
  - Open the Cloud Manager or API Manager user interface.
  - On the welcome page, click the Download Toolkit tile. The two toolkit options are listed.
  - Click your platform alongside either the CLI option or the CLI + LoopBack + Designer option as required, then save the associated compressed file to your local file system.
  - To download the API Designer credentials (Client ID and Client Secret), click Download alongside API Designer Credentials, then save the `designer_credentials.json` file to your local file system. A command is provided for installing the credentials, as described in detail in [step 6](#).
  - To download the toolkit credentials (Client ID and Client Secret), click Download alongside Credential, then save the `credentials.json` file to your local file system. A command is provided for installing the credentials, as described in detail in [step 5](#).
  - Close the Install API Connect Toolkit window.

### 2. Extract the contents of the toolkit compressed file to a folder of your choice.

The contents of the file depend on the your chosen toolkit option and platform, as follows:

- The `apic-slim` or `apic-slim.exe` file is the CLI for IBM API Connect.
- The `apic` or `apic.exe` file is the CLI for IBM API Connect including LoopBack support.
  - Tip: If you are using the CLI option, then if you rename the `apic-slim` file to `apic`, or the `apic-slim.exe` file to `apic.exe`, you can run the CLI commands exactly as documented, copy and paste sample commands from the documentation, and use any command scripts as-is if you later move to the CLI + LoopBack + Designer option.
- The `api_designer-platform` file is the API Designer user interface application for the specified platform.

### 3. Optional: Delete the `$HOME/.apiconnect/node_cli` directory.

You need to do this only if you replaced a version of `apic` or `apic.exe` and are using the `apic lb4` command. You need to delete the directory `get apic` to unpack the new loopback.

On Windows, `$HOME` is defined by environment `USERPROFILE`.

### 4. Run the CLI.

- For the Mac OSX or Linux® platforms, complete the following steps:
  - Open a terminal instance and navigate to the folder where you extracted the contents of the toolkit compressed file.
  - Make the CLI file an executable file by entering the following command:

```
chmod +x download_name
```

Where `download_name` is the name of the toolkit file that you downloaded, either `apic` or `apic-slim`.

- Run CLI commands as follows:

```
./apic command_name_and_parameters
```

or

```
./apic-slim command_name_and_parameters
```

For details of the CLI commands, see [apic](#).

- For the Windows platform, complete the following steps:
  - Open a command prompt and navigate to the folder where you extracted the contents of the toolkit compressed file.
  - Run CLI commands as follows:

```
apic command_name_and_parameters
```

or

```
apic-slim command_name_and_parameters
```

For details of the CLI commands, see [apic](#).

Tip: Add the folder location of your CLI file to your `PATH` variable so that you can run CLI commands from anywhere in your file system.

### 5. Install the toolkit credentials.

If you do not install the toolkit credentials that are provided for download, as detailed previously in the [toolkit credentials download instructions](#), API Connect uses a default set of credentials that are identical for all deployments. However, the downloaded credentials were generated during the deployment of your API Connect system and are unique to your installation. To install the credentials into your local toolkit, run the following command:

```
apic client-creds:set toolkit_credentials_file_path/credentials.json
```

where `credentials_file_path` is the location to which you downloaded the toolkit credentials JSON file. After you have run this command, your toolkit uses these new credentials to authenticate with the management server.

Note: At any one time, you can use only one set of toolkit credentials for login to a management server from the toolkit CLI. If you want to log in to a different management server you must install the toolkit credentials from that management server.

To revert to the default toolkit credentials for all login operations from the toolkit CLI, use the following command:

```
apic client-creds:clear
```

For increased security, an administrator can remove the default credentials from the management server by completing the following steps:

- Log in to the management server as an administrator; see [Logging in to a management server](#).
- Delete the default credentials by running the following commands:

```
apic registrations:delete toolkit --server mgmt_endpoint_url
apic registrations:delete consumer-toolkit --server mgmt_endpoint_url
```

#### 6. Install the API Designer credentials.

If you do not install the API Designer credentials that are provided for download, as detailed previously in the [API Designer credentials download instructions](#), API Connect uses a default set of credentials that are identical for all deployments. However, the downloaded credentials were generated during the deployment of your API Connect system and are unique to your installation. To install the custom credentials into your local API Designer, set the `APIC_DESIGNER_CREDENTIALS` environment variable to the credentials download location, using the mechanism appropriate for your operating system.

After you have set the `APIC_DESIGNER_CREDENTIALS` environment variable, your API Designer uses the new credentials to authenticate with the management server.

- Windows:

If you are using Windows, create an environment variable using one of the following methods:

- Create a permanent environment variable so that you can start API Designer from any location on your computer:
  - Open the Environment Variables page: Click Start, Settings, System and in the "Related Settings" section of the page, click Advanced System Settings.
  - On the Advanced tab of the System Properties dialog box, click Environment Variables.
  - In the "User variables" section, click New and create an environment variable with the following settings:
    - Variable: `APIC_DESIGNER_CREDENTIALS`
    - Value: `<designer_credentials_file_path>\designer_credentials.json` where `<designer_credentials_file_path>` is the location where you stored the `designer_credentials.json` file.
  - Click OK to save the new environment variable, and then exit the dialog box.
- If you don't have the required permissions to create a permanent environment variable, you can create a temporary one that will only be used while you are running the API Designer application. The following steps must be performed every time you start API Designer:
  - Open the Windows command prompt.
  - Run the following command to set the temporary environment variable:

```
set APIC_DESIGNER_CREDENTIALS=<designer_credentials_file_path>\designer_credentials.json
```

where `<designer_credentials_file_path>` is the location where you stored the `designer_credentials.json` file.

Note: Leave the command prompt open for the next step. You must set the temporary variable and start the API Designer within the same Windows command session.

- Now run the following command to start API Designer:

```
C:\>"Program Files\API Designer\API Designer.exe"
```

By default, API Designer is installed in the `C:\Program Files\API Designer` folder as shown in the example. If you installed it into a different location, use your own location in the command. Note that the path and file name are enclosed in quotation marks because they contain spaces.

- Mac:

If you are using Mac OS X, create an environment variable using one of the following methods:

- Run the following command to set the global environment variable:

```
launchctl setenv APIC_DESIGNER_CREDENTIALS <designer_credentials_file_path>/designer_credentials.json
```

where `<designer_credentials_file_path>` is the location where you stored the API Designer credentials JSON file.

- Pass in the environment variable while starting API Designer from the command line. With this method, you must run the following command every time you start API Designer:

```
APIC_DESIGNER_CREDENTIALS=<designer_credentials_file_path>/designer_credentials.json open
<designer_application_file_path>/API Designer.app'
```

where:

- `<designer_credentials_file_path>` is the location to which you downloaded the API Designer credentials JSON file.
- `<designer_application_file_path>` is the location to which you downloaded and uncompressed the API Designer application.

#### 7. Start the API Designer application from the location to where you stored the extracted file.

Note:

- To uninstall the API Designer application on a Windows platform with a non Administrator account, complete the following steps:
  - In Windows File Explorer, navigate to the `USER_HOME\AppData\Local\Programs\api-designer` folder.
  - Run the **Uninstall API Designer application** application. Do **not** use the **Add or remove programs** window.
- To uninstall the API Designer application on a Windows platform with an Administrator account, you can either run the **Uninstall API Designer application** application, or you can use the **Add or remove programs** window.

## Results

The IBM API Connect toolkit CLI and, if selected, the API Designer user interface application are installed on your local system.

For information on using the API Designer user interface, see [Developing your APIs and applications](#).

---

## Logging in from API Designer

To log in from the API Designer user interface for IBM® API Connect, you will need the host name of the management server and the user name and password associated with your API Manager account.

---

### About this task

API Designer provides a secure login to a management server for staging and publishing APIs and Products. You log in to the same host that was configured for API Manager and use your API Manager user name and password. You will need an account on API Manager and membership in a provider organization to access the API Designer.

You can also use the Explorer tab to see how your APIs look to a consumer in the Developer Portal. You can check the descriptions of the different artifacts, and refine any schemas or examples, and you can also use the Try it tool to test the behavior of the API.

Note:

- Configuration changes that are made in API Manager are not available in API Designer concurrently. You will need to log out of API Designer and log back in to access any updates.
- Set `SKIP_IBMID_LOGIN=true` as an environment variable to bypass the IBM ID login screen for API Designer. This applies to users who are working on an internal network, without access to external networks.

---

### Procedure

To log in to API Designer, complete the following steps:

1. Download and extract API Designer, as described in [Installing the toolkit](#).
  2. Start the API Designer user interface by running the application from the location where you extracted the contents of the toolkit compressed file.  
Note: If you see a message stating that credentials cannot be found, download new credentials as explained in [Downloading the toolkit](#).
  3. Optional: If you see the Accept License page, click the URL read the license agreement, then click the button to accept the license. This prompt appears only one time. It does not appear once you have accepted the license on the current system.
  4. Optional: If you see the IBM ID login page, enter your IBM customer ID.  
Note: You won't receive the IBM ID page if you are already authorized, or if you have set the `SKIP_IBMID_LOGIN` environment variable to `true`.
  5. Open a directory on your local file system to store the API and Product specifications that you create in API Designer. The system defaults to a previously-accessed directory.  
If you want to switch to a different directory after logging in, click Switch directory on the menu bar in the header.  
  
Note: If you want to use an OpenAPI definition file from elsewhere, downloaded from an external website for example, rather than created by using API Connect, don't copy the file into your API Designer directory.  
Such an API will **not** be visible in the API Designer user interface if it doesn't contain the API Connect specific sections that are required by API Designer.  
  
Instead, copy the OpenAPI definition file to another folder, then import it into API Designer; the import operation adds the necessary API Connect specific sections; see [Adding a REST API by importing an OpenAPI definition file](#).  
  
For more information on the API Connect specific sections in an OpenAPI definition file, see [IBM extensions to the OpenAPI specification](#).
  6. If this is the first time you have logged in, establish a connection to API Connect by completing the following steps:
    - Contact your administrator and request the "management endpoint URL" for connecting to API Connect.  
The management endpoint URL looks similar to the following example: `https://<host.domain>` and indicates the location where the API Manager is running.  
  
Note: If you are using API Connect as a component of Cloud Pak for Integration, the management endpoint URL follows this format:  
`https://cpd-your-company.domain/integration/apis/APIC_namespace/APIC_instance`  
  
Example: `https://cpd-company.com/integration/apis/apic/production`  
  
Do not add `/manager` to the URL. If you include `/manager` at the end of this URL, the connection with Cloud Pak for Integration will fail with a message stating that the host URL is invalid.
    - In the dialog window, enter the management endpoint URL as the Host URL and click Next.
    - Type the user name and password for your API Manager account.
    - Select a User Registry from the list.
    - Click Sign In.
  7. For subsequent log ins, select the appropriate cloud where you will be working. You can select from any previous cloud connections.
  8. When more than one cloud connection is configured, you can switch clouds when developing APIs and Products. See [Switching clouds from API Designer](#).  
Note: If you are using an OIDC user registry that supports single sign-on (SSO), you cannot switch clouds to log in to the same user registry with a different user ID. You must first restart the API Designer application so that you are logged out of the registry, then select the required cloud connection.
  9. Click Add Another Cloud to add another cloud connection. Enter the URL, user name and password, and user registry to add the cloud connection.  
Note: If you are using an OIDC user registry that supports single sign-on (SSO), and you are already logged in, you cannot add a new cloud connection to log in to the same user registry with a different user ID. You must first restart the API Designer application so that you are logged out of the registry, then add the new cloud connection.
  10. Delete a cloud connection by selecting the trash can icon.
  11. API Designer will launch with the organization set to your provider organization in the selected cloud. If you have accounts for multiple provider organizations, it will default to the first organization in alphabetical order.
  12. To choose another provider organization, select it from the drop-down list.
- [Setting the locale for API Designer](#)  
How to set one of the supported locales when you start the API Designer application.

- [Switching clouds from API Designer](#)  
You can switch clouds to work in another cloud in API Designer.
- [Working offline with API Designer](#)  
You can work offline (without a connection to the cloud) with API Designer to create APIs. Some functionality is missing due to the lack of a cloud connection.
- [Gathering support information in API Designer](#)  
When you report an issue with the API Designer application, enable debug mode and provide as much information as possible.

## Related information

---

- [Activating your API Manager user account](#)
- [Creating a provider organization account](#)

## Setting the locale for API Designer

How to set one of the supported locales when you start the API Designer application.

### About this task

---

If you want to use a different locale than your current operating system (OS) locale, you can set a locale code by using the `--lang` argument when you start the API Designer application. This locale is then effective for the duration that the application is open. If no `--lang` argument is set, then the current OS locale is used.

### Procedure

---

To change the locale for API Designer, complete the following instructions.

From the command line, run the following command to start the API Designer application with the `--lang` argument:

**Mac OS X** Mac:

```
open "API Designer.app" --args --lang=locale_code
```

**Windows** Windows:

```
"API Designer".exe --args --lang=locale_code
```

Where *locale\_code* is one of the following supported locale codes:

Table 1. List of supported locale codes and their associated language

Locale code	Language
cs	Czech
de	German (Standard)
en-US	English (United States)
es	Spanish
fr	French (Standard)
it	Italian (Standard)
ja	Japanese
ko	Korean
nl	Dutch (Standard)
pl	Polish
pt-br	Portuguese (Brazil)
ru	Russian
tr	Turkish
zh-cn	Chinese (PRC)
zh-tw	Chinese (Taiwan)

For example, running the following command on a Mac:

```
open "API Designer.app" --args --lang=es
```

sets the API Designer language to Spanish.

### Related tasks

---

- [Installing the toolkit](#)

## Switching clouds from API Designer

You can switch clouds to work in another cloud in API Designer.

### About this task

---

API Designer allows the flexibility to connect to multiple clouds. For example, your organization may have separate clouds for testing, development, and demos. You select the cloud connection when you log in, but you can also switch clouds when you are developing APIs and Products. Multiple clouds must be configured in order to switch clouds. You must have a valid API Manager account to connect to a cloud.

Note: If you are using an OIDC user registry that supports single sign-on (SSO), you cannot switch clouds to log in to the same user registry with a different user ID. You must first restart the API Designer application so that you are logged out of the registry, then select the required cloud connection.

---

## Procedure

To switch clouds for working in API Designer, complete the following steps:

1. Log in to API Designer and select a cloud, or configure a new cloud connection.
2. From the Develop: APIs and Products screen, you can switch to another cloud by selecting it from the Switch Cloud Connection menu.
3. Delete a cloud connection by selecting the trash can icon.

---

## Related information

- [Activating your API Manager user account](#)
- [Creating a provider organization account](#)

---

## Working offline with API Designer

You can work offline (without a connection to the cloud) with API Designer to create APIs. Some functionality is missing due to the lack of a cloud connection.

---

### About this task

API Designer provides the capability to work in offline mode without a connection to a management server. You can work on APIs with no management server, but some functions will be changed or missing.

The preferred method for working in API Designer is with a Cloud Connection to the management server when logging into API Designer. Working while connected to a management server provides full functionality, with automatic field completion. When working offline, you will need to remember the values to complete the fields that are backfilled from the management server.

Following are the differences when working in offline mode:

- There is no connection to a management server.
- There is no Provider Organization selection.
- Only the Develop APIs and Products tile may be selected. The Learn more and Connect menu tiles are not available.
- When creating an API, the Gateway Type defaults to V5 Gateway and the default policies are loaded.
- Stage and Publish for APIs and Products functions are not available.
- The Sandbox catalog is not available.
- When configuring a Security Definition, there are no selection lists available for the User Registry for a Basic Auth security definition or for the OAuth Provider for an OAuth 2 security definition. You will need to enter the User Registry and OAuth Provider values correctly as text.
- When adding custom Visibility and custom Subscribability values for a product, there is no selection list for the Organization field. You will need to enter the organization name correctly as text.

---

## Procedure

To work in offline mode in API Designer, complete the following steps:

1. Log in to API Designer and open a directory location.
2. To start offline mode, choose Work Offline from the selector in the upper right corner of the Connect to Cloud screen.
3. The Home page appears with the Develop APIs and Products tile. Select Develop APIs and Products to start working in offline mode.
4. To return to online mode, choose Connect to Cloud in the upper right hand corner and select the management server to log in to.

---

## Gathering support information in API Designer

When you report an issue with the API Designer application, enable debug mode and provide as much information as possible.

To report an issue with API Designer, enable debug mode, collect the recommended information, and submit it with your report. Including the information will help IBM resolve the problem more quickly.

- [Enabling debug mode on Mac](#)
- [Enabling debug mode on Windows](#)
- [Enabling debug mode on Linux](#)
- [Information to gather for support](#)

---

## Enabling debug mode on Mac

Start API Designer with debug mode enabled:

1. Start a terminal or command prompt.
2. Navigate to the directory where API Designer is installed.



3. Run the following command to start API Designer in debug mode:

```
DESIGNER_DEBUG=true open API/Designer.app/
```

If you add `DESIGNER_DEBUG=true` to your `.bash_profile`, then every time you start API Designer it will open in debug mode.

## Enabling debug mode on Windows

Add a Windows environment variable that enables debug mode for API Designer:

1. Type "Environment" in the Windows search bar (on the task bar) and select Edit environment variables for your account.
2. In the Environment Variables dialog box, click New in the "User variables" section.
3. In the New User Variable dialog box, add the following variable settings and then click OK:
  - Variable name: `DESIGNER_DEBUG`
  - Variable value: `True`
4. Click OK to save your change.


## Enabling debug mode on Linux

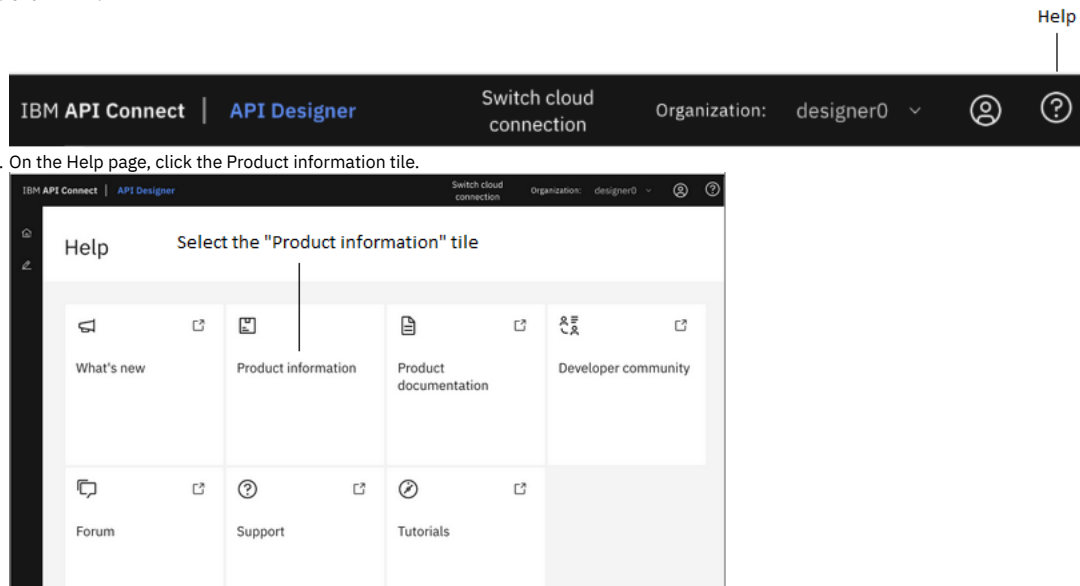
To start API Designer with debug mode enabled, add the following command to your `.bash_profile`:

```
EXPORT DESIGNER_DEBUG=true
```

## Information to gather for support

Gather the following information about your problem and include it with your report.



- Operating system where you are running API Designer (Mac, Windows, or Linux)
- Product versions for API Designer and the API Manager that is connects to:
  1. Open API Designer.
  2. Click .



3. On the Help page, click the Product information tile.

4. On the Product information page, note down the following values for the following labels on the page:
  - Connected to API Manager Product version
  - Package

API Manager version	<b>Product information</b> <hr/> <b>Connected to API Manager Product version</b> 10.0.3.0
API Designer package version	<b>Git information</b> <hr/> <b>Git</b> 9affdfeeb3855c194edabdb933e1ee3c6f2afddb <b>Package</b> 1.0.0 <b>api-client</b> 0.2.3 <b>apim-sdk</b> 5.0.31 <b>assembly-debugger</b> 0.1.38 <b>async-api-editor</b> 1.0.5 <b>doc-types</b> 0.1.20

- HAR file  
The HAR file contains the information about your browser's interaction with the web, and can be helpful in determining why your problem is happening. Complete the following steps to export the HAR file from the browser where you use API Designer:
  1. Open API Designer.
  2. Open the browser's developer console; for example in Google Chrome, click  More tools > Developer tools.
  3. In the developer console, click the Network tab.
  4. Click  to download the HAR file.
- Are you using API Designer in Online mode or Offline mode?  
If you're using Online mode, are you connecting to a local test environment, or to an API Manager server?
- When does the problem happen?  
Provide reproducible steps with optional screen captures or video so the support team can recreate the problem.
- Supporting data.  
Provide any related data that would be useful while examining the problem; for example, a copy of the API or Product that you were using.
- How often does the problem happen?  
Is the problem intermittent? Does it happen constantly? Does it only happen with certain artifacts (for example, only with a particular API)? If the problem is associated with an artifact, try to include it with your report.

## Using the developer toolkit command-line tool

The IBM® API Connect developer toolkit provides a command-line tool, **apic**, that you can use to perform all API Connect tasks. You can also use the command-line tool to script tasks such as continuous integration and delivery.

The command line tool is described in detail in the following subtopics. For summary reference material for each of the available commands for the developer toolkit, see [Command-line tool reference for the developer toolkit](#).

- [Overview of the command-line tool](#)  
The IBM API Connect developer toolkit provides commands for cloud administration and API development and management.
- [Logging in to a management server](#)  
You log in to a management server from the command line by using the **apic login** command. The parameters that you supply determine the identity provider that is used to authenticated the supplied user ID, and the scope of the tasks that can be performed after successful log in.
- [Logging in to a management server with an OIDC registry](#)  
Logging in with an OIDC registry involves several steps, in which you obtain a temporary token and pass it to the toolkit to ensure it can communicate securely with the management server.
- [Cloud administration commands](#)  
A summary of the cloud administration commands in the IBM API Connect developer toolkit.
- [API development and management commands](#)  
A summary of the core commands in the IBM API Connect developer toolkit.
- [Creating APIs and applications](#)  
You can develop API proxies and API implementations by using the developer toolkit. In the documentation, *API* refers to the API proxy and *application* refers to the API implementation.
- [Publishing APIs and applications](#)  
To publish APIs and applications by using the developer toolkit of IBM API Connect, you set configuration variables to define where you want to publish, log in to the target cloud platform, and then use the appropriate publishing commands.
- [Managing API Products](#)  
Use the **apic products** and **apic apis** commands to manage Products and APIs that have been published to IBM API Connect Catalogs. Use the **--scope space** option to manage Products and APIs that have been published to Spaces within Catalogs.

- [Working with Drafts](#)  
Co-locate your APIs and applications in your local source code control systems to support typical development activities such as commits, branching, merges, continuous integration, and so on. The developer toolkit provides the bridge from the developer's environment to the IBM API Connect runtime services.
- [Reading input from the command line](#)  
If a developer toolkit command takes a file as an input parameter, you can direct the command to read the input directly from the command line rather than supplying a separate file; this can be useful when writing scripts to automate command line operations, for example.
- [Scripting with the toolkit commands](#)  
The IBM API Connect developer toolkit provides commands for cloud administration and API development and management.
- [Working with OpenAPI extensions](#)  
Use the developer toolkit CLI **extensions** commands to manage OpenAPI extensions in your Catalogs or Spaces.

---

## Overview of the command-line tool

The IBM® API Connect developer toolkit provides commands for cloud administration and API development and management.

### Command syntax

---

In general, commands have the following syntax:

```
apic command:sub-command [argument] [options]
```

where

- *command* is the command, usually the thing on which you are acting (for example, product, app, API, Catalogs, and so on).
- *sub-command* is the action to perform.
- *argument* is the argument, where applicable (for example, `catalog`).
- *options* are any number of command-line options, which have the form `--option [value]`. Options also have a short form with a single dash instead of a double dash.

For example, `apic apps:publish --server mgmthost.com`.

For some commands, either the command or sub-command portion is optional. For example:

- `apic products:publish` is equivalent to `apic publish`.
- `apic products:list` is equivalent to `apic products`.

The `create` command has a slightly different syntax:

```
apic create:type [options]
```

Use the `-h` or `--help` option to view command help.

Note: The language in which the CLI help text, and other command response text, is displayed is determined by the locale setting on your local machine.

### Viewing command line tool help

---

Display general command-line help information by entering the following command: `apic --help` or `apic -h`. Display help information for a specific `apic` command by entering the following command: `apic command_name --help` or `apic command_name -h`.

### Viewing version information

---

Display the version of the command-line tool by entering the command: `apic --version`.

### Setting the mode

---

The API Connect toolkit CLI runs commands on either the provider organization or on the consumer organization. When you run the command, you use the different mode options to identify the instance on which you want to run the command. The following list provides a summary of the modes that are available:

apim

This mode applies to the administrative instances of the CLI. When you run commands in this mode, it uses the overall API Manager as its scope. This mode is the default mode.

An example of running a command in the apim mode:

```
apic extensions:list-all --mode apim --scope catalog --server server1 --catalog catalog1 --org myOrg --configured-gateway-service service1
```

portaladmin

This mode applies to the developer portal instances of the CLI. This mode supports commands that you can use to replicate all or part of your Developer Portal environment.

An example of running a command in the portaladmin mode:

```
apic --mode portaladmin custom-theme:create-export --catalog catalog1 --org myOrg --server server1 --format json
```

consumer

This mode applies to the consumer instances of the CLI. This command uses the consumer organizations that you have permission to access as its scope. You must specify this mode or set it as the default value for the command to use it.

An example of running a command in the consumer mode:

```
apic catalogs:list --mode consumer --scope catalog --server server1 --catalog catalog1 --org myOrg
```

To avoid having to supply the parameter on every consumer CLI command, you can set the `mode` configuration variable, as described in the next section.

## Using configuration variables

You can set the values of commonly-used properties in configuration variables. In general, it's easier and more consistent to set configuration variables instead of specifying them using command-line options.

Note:

You can set a configuration variable locally (the default) to affect only the current LoopBack project, or globally (with command-line option `-g`), to affect all projects. The local value supersedes the global value. You can set local configuration variables only for LoopBack projects. When you set configuration variables for OpenAPI projects, they are always global.

The values of local configuration variables are stored in the `project-root/.apiconnect/config` file, where `project-root` is the project root directory. The values of global configuration variables are stored in the `user-home-dir/.apiconnect/config` file, where `user-home-dir` is the user's home directory.

Use the following commands to work with configuration variables:

- `apic config:get varname` - Get a configuration variable. Use `apic config` to display the values of all local configuration variables or `apic config -g` to display the values of all global configuration variables.
- `apic config:set varname` - Set or update the specified configuration variable.
- `apic config:delete varname` - Delete the specified configuration variable.
- `apic config:clear` - Delete all configuration variables.

You set configuration property values by using the `apic config:set` command. By setting configuration properties (for example `catalog` and `app`), you do not need to supply values for these options when you enter a command.

Additionally, you can use `apic properties` commands to work with configuration properties:

- `apic properties:clear` - Clear the configuration properties.
- `apic properties:create` - Augment the configuration properties with additional name/value pairs.
- `apic properties:delete` - Delete the configuration property.
- `apic properties:get` - Get the configuration property.
- `apic properties:list` - List the configuration properties.
- `apic properties:update` - Update the configuration property.

Note:

If you have an environment variable of the same name as a CLI configuration property then, by default, its value will override the value of the corresponding CLI configuration property for any CLI command at that scope.

For example, if you have defined an environment variable called `SPACE` then, by default, that value will be assumed for the value of the `--space` parameter in the following command, regardless of any `space` configuration property setting:

```
apic products:publish my_product.yaml --scope space
```

To prevent environment variables overriding CLI configuration properties, define an environment variable called `APIC_LOAD_FROM_ENV`, set to the value `false`.

The following table describes the configuration variables:

Table 1. Configuration variables

Variable name	Description	Use instead of (or override with) these flags...
catalog	Default Catalog URI for all commands that manage aspects of a Catalog. Form: <code>mgmt-server/api/catalogs/org-name/catalog-name</code> , where <code>mgmt-server</code> is the management server, <code>org-name</code> is the organization name, and <code>catalog-name</code> is the Catalog name.  Note: The Catalog name <code>apic-dev</code> is reserved for local testing.	--catalog, --organization, --server
cloud	Default management server host name for cloud administration commands. Form: <code>mgmt-server/api/</code> .	--server
consumer	Default URI of an API consumer. Form: <code>mgmt-server/api/consumer-orgs/org-name/catalog-name/consumer-org-name</code> , where <code>mgmt-server</code> is the management server, <code>org-name</code> is the organization name, <code>catalog-name</code> is the Catalog name, and <code>consumer-org-name</code> is the consumer organization name.	--server, --organization, --catalog, --consumer
mode	The default value of the <code>--mode</code> parameter for CLI commands. Set the value to <code>apim</code> or <code>consumer</code> depending on whether you want to run commands on a provider organization or a consumer organization. If you do not set this variable, and do not supply a <code>--mode</code> parameter on a command, the value <code>apim</code> is assumed.	--mode
org	Default org URI for all commands that manage organizations. Form: <code>mgmt-server/api/orgs/org-name</code> , where <code>mgmt-server</code> is the management server, <code>org-name</code> is the organization name.  The <code>mgmt-server</code> portion sets the default value of the <code>--server</code> option.  You can append the port number to the server name if it is not the default value of 443.	--organization, --server
space	Default Space URI for all commands that manage aspects of a Space. Form: <code>mgmt-server/api/spaces/org-name/catalog-name/space-name</code> , where <code>mgmt-server</code> is the management server, <code>org-name</code> is the organization name, <code>catalog-name</code> is the Catalog name, and <code>space-name</code> is the Space name.  You can append the port number to the server name if it is not the default value of 443.	--server, --organization, --catalog, --space

To set configuration properties, enter the following command:

```
apic config:set name=value
```

where `name` is the name of the configuration property and `value` the value to assign to it.

For example:

```
apic config:set catalog=https://platform-api.myserver.com/api/catalogs/climbon/sandbox
```

## Scripting commands

---

It's often helpful to automate a series of `apic` commands in a shell script. Since the `apic` tool first requires you to interactively accept the license, you must first use the following command:

```
apic --accept-license
```

Once you do that, your scripts can run non-interactively.

To disable collection of usage analytics, enter this command:

```
apic --live-help
```

## Related information

---

- [Command-line tool reference for the developer toolkit](#)

---

## Logging in to a management server

You log in to a management server from the command line by using the `apic`

`login` command. The parameters that you supply determine the identity provider that is used to authenticate the supplied user ID, and the scope of the tasks that can be performed after successful log in.

Attention: If you want to log in with an OIDC user registry or if you are an IBM Cloud Pak for Integration (CP4I) user, see [Logging in to a management server with an OIDC registry](#).

1. Clear the credentials from any previous login by running the following command:

```
apic client-creds:clear
```

Clearing the credentials ensures that you do not inadvertently log in with the wrong set of credentials (for example, from a different product release or environment).

2. Set the credentials that you want to use now by running the following command:

```
apic client-creds:set path_to_credentials/credential.json
```

where `path_to_credentials` is the location of the `credential.json` file that you want to use. For example:

```
apic client-creds:set /Users/local_user/credential.json
```

3. Log in by running the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm realm
```

The parameters for the `apic login` command are as follows:

*mgmt\_endpoint\_url*

The URL depends on the type of organization that you are logging in with:

- Cloud admin organization or Provider organization: Use one of the following URLs:
  - platform API endpoint URL
  - management API manager URL
- Consumer organization: Use the consumer API endpoint URL

These endpoint URLs are configured during the installation of API Connect. If you have access to the Cloud Manager user interface, you can view the configured endpoint URLs as described in [Viewing platform and UI endpoints](#), ignoring any segments at the end of the displayed URLs. If you are not sure of the endpoint URL, ask your administrator.

*user\_id*

The user ID you want to log in with. Depending on the tasks that you want to perform, this user ID might be any of the following:

- A user ID that is a member of the cloud administration organization. This is an ID that you could also use to log in to the Cloud Manager user interface.
- A user ID that is a member of a provider organization. This is an ID that you could also use to log in to the API Manager user interface.
- A user ID that is a member of a consumer organization. This is an ID that you could also use to log in to the Developer Portal.

*password*

The password associated with the supplied user ID.

*realm*

The `realm` parameter specifies the identity provider that is used to authenticate the supplied user ID, and the scope of the tasks that can be performed after successful log in.

The format of the `realm` depends on the type of user, as follows:

- Member of the cloud administration organization:

```
admin/identity_provider
```

To determine the identity provider, see [How to determine the identity provider](#).

- Member of a provider organization:

```
provider/identity_provider
```

To determine the identity provider, see [How to determine the identity provider](#).

- Member of a consumer organization:

```
consumer:provider_org:catalog/identity_provider
```

where *provider\_org* is the name of your provider organization, and *catalog* is the name of the Catalog in that provider organization. To determine the identity provider, see [How to determine the identity provider](#).

Important:

- If you have installed toolkit credentials, as detailed in [Installing the toolkit](#), then at any one time, you can use only one set of toolkit credentials for login to a management server from the toolkit CLI. If you want to log in to a different management server you must install the toolkit credentials from that management server.

To revert to the default pre-supplied toolkit credentials for all login operations from the toolkit CLI, use the following command:

```
apic client-creds:clear
```

- If you log in to the CLI as a member of a consumer organization, you must supply the `--mode=consumer` parameter to the `apic login` command, and to all consumer commands. To avoid having to type the parameter every time, you can set the `mode` configuration variable, by entering the following command:

```
apic config:set mode=consumer
```

You can also use the command interactively; enter `apic login` and you will be prompted for the values. For example:

```
apic login
Enter your API Connect credentials
Server? platform-api.myserver.com
Realm? provider/default-idp-2
Username? myuser
Password?
Logged into myserver.com successfully
```

Note: If you see a message stating that credentials cannot be found, download new credentials as explained in [Downloading the toolkit](#).

## How to determine the identity provider

If you want to log in as a member of the cloud administration organization, or as a member of a provider organization, you can help determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope scope --server mgmt_endpoint_url --fields title,realm
```

where *scope* has the value `admin` or `provider` depending on whether you want to log in as a member of the cloud administration organization, or as a member of a provider organization. The output lists the names and titles of all identity providers, for example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`, and the default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

By default, API Connect creates a local user registry for user login for every context. The identity providers associated with these default registries are as follows:

Registry	Identity provider name
Cloud Manager Local User Registry (for login as a member of the cloud administration organization)	default-idp-1
API Manager Local User Registry (for login as a member of a provider organization)	default-idp-2
Sandbox Catalog User Registry (for login as a member of a consumer organization)	sandbox-idp

If you want to log in as a member of a consumer organization, and you are not using the default Sandbox Catalog User Registry, ask your administrator for the name of your identity provider.

## Logging out

To log out of a management server, use the following command:

```
apic logout --server mgmt_endpoint_url
```

## Logging in to a management server with an OIDC registry

Logging in with an OIDC registry involves several steps, in which you obtain a temporary token and pass it to the toolkit to ensure it can communicate securely with the management server.

Attention: Logging in with an OIDC registry is supported for the developer toolkit (used with API Manager) and the Developer Portal admin toolkit, but is not supported for members of consumer organizations. If you are using IBM Cloud Pak for Integration (CP4I), review this topic for login instructions. If you want to log in using a non-OIDC registry, see [Logging in to a management server](#).

To log in, complete the following steps:

1. Clear the credentials from any previous login by running the following command:

```
apic client-creds:clear
```

Clearing the credentials ensures that you do not inadvertently log in with the wrong set of credentials (for example, from a different product release or environment).

2. Set the credentials that you want to use now by running the following command:

```
apic client-creds:set path_to_credentials/credential.json
```

where `path_to_credentials` is the location of the `credential.json` file that you want to use. For example:

```
apic client-creds:set /Users/local_user/credential.json
```

3. Log in by running the following command:

```
apic login --server mgmt_endpoint_url --sso
```

where `mgmt_endpoint_url` is the endpoint URL. When you log in with a Cloud admin or Provider organization, specify one of the following URLs:

- platform API endpoint URL
- management API manager URL

These endpoint URLs are configured during the installation of API Connect. If you have access to the Cloud Manager user interface, you can view the configured endpoint URLs as described in [Viewing platform and UI endpoints](#), ignoring any segments at the end of the displayed URLs. If you are not sure of the endpoint URL, ask your administrator.

CP4I: If you are using API Connect as a component of IBM Cloud Pak for Integration (CP4I), your administrator can provide the correct URL.

4. When the toolkit prompts for the context, type **admin** API Connect (administrators) or **provider** (everyone else):

```
Context? provider
```

5. The server responds with the following message:

```
Please copy and paste the url https://mgmt_endpoint_url/auth/manager/sign-in/?from=TOOLKIT to a browser to start the authentication process.
Do you want to open the url in default browser? [y/n]: y
```

Take one of the following actions:

- If you want to use your default browser to log in to API Manager, type **y** and press Enter. The API Manager sign-in page opens in a new browser tab.
- If you don't use to use your default browser, type **n** and press Enter.
  - Copy the URL from the command window.
  - Open a browser, paste the URL, and press Enter. The API Manager sign-in page displays.

6. On the API Manager sign-in page, select the OIDC registry and then log in to API Connect.

CP4I: Select the Common Services User Registry.

After you log in, API Connect displays the "You are authenticated!" message and provides a temporary token. Copy the token.

7. Return to the command window. Paste the token at the **API Key?** prompt and press Enter.

When the token is validated and you are successfully logged in to the toolkit, the following message displays:

```
Warning: Using default toolkit credentials.
Logged into mgmt_endpoint_url successfully
```

Note: If you see a message stating that credentials cannot be found, download new credentials as explained in [Downloading the toolkit](#).

## Logging out

---

To log out of a management server, use the following command:

```
apic logout --server mgmt_endpoint_url
```

## Cloud administration commands

---

A summary of the cloud administration commands in the IBM® API Connect developer toolkit.

## Authenticating

---

Use the `apic login` command to authenticate to an API Manager service, and the `apic logout` command to remove your local authentication credentials.

Note: When you authenticate successfully, your credentials are stored, in plain text, in the file `Linux .netrc` or `Windows _netrc`. You should therefore set the file permissions in such a way that your credentials are not accessible by others.

For full details on how to log in to your management server from the CLI, see [Logging in to the management server](#).

## Configuring the command-line tool to use TLS certificates

---

API Manager uses TLS profiles to secure data transmission. For information on how to create a TLS profile in API Manager, see [TLS profiles](#).

To configure the toolkit command-line tool to use certificates to communicate with an API Manager that has TLS profiles enabled, follow these steps:

For more information about the `NODE_EXTRA_CA_CERTS` environment variable, see [Node.js documentation](#).

Note: When using any CLI command that applies directly to the configuration of your IBM API Connect cloud, you must supply the parameter setting `--org admin` for a command that requires an organization name as a parameter.

## Command summary

The following tables summarize **apic** commands for cloud administration.

Table 1. Summary commands for cloud administrators

Command	Description	Sub-commands
<b>apic analytics-services</b>	Analytics Service commands	<ul style="list-style-type: none"> <li>• <b>apic analytics-services</b> - Analytics Service collection operations.</li> <li>• <b>apic analytics-services:clear</b> - Clear the Analytics Services.</li> <li>• <b>apic analytics-services:create</b> - Create a Analytics Service.</li> <li>• <b>apic analytics-services:delete</b> - Delete the Analytics Service by name or ID.</li> <li>• <b>apic analytics-services:get</b> - Get the Analytics Service by name or ID.</li> <li>• <b>apic analytics-services:list</b> - List the Analytics Services.</li> <li>• <b>apic analytics-services:update</b> - Update the Analytics Service by name or ID.</li> </ul>
<b>apic associates</b>	Associates commands	<ul style="list-style-type: none"> <li>• <b>apic associates</b> - Associate collection operations.</li> <li>• <b>apic associates:get</b> - Get the Associate by name or ID.</li> <li>• <b>apic associates:list</b> - List the Associates.</li> </ul>
<b>apic availability-zones</b>	Availability Zone commands	<ul style="list-style-type: none"> <li>• <b>apic availability-zones</b> - Availability Zone collection operations.</li> <li>• <b>apic availability-zones:clear</b> - Clear the Availability Zones.</li> <li>• <b>apic availability-zones:create</b> - Create a Availability Zone.</li> <li>• <b>apic availability-zones:delete</b> - Delete the Availability Zone by name or ID.</li> <li>• <b>apic availability-zones:get</b> - Get the Availability Zone by name or ID.</li> <li>• <b>apic availability-zones:list</b> - List the Availability Zones.</li> <li>• <b>apic availability-zones:update</b> - Update the Availability Zone by name or ID.</li> </ul>
<b>apic cloud-settings</b>	Cloud settings commands	<ul style="list-style-type: none"> <li>• <b>apic cloud-settings:get</b> - Get the Cloud Setting.</li> <li>• <b>apic cloud-settings:mail-server-configured</b> - indicates whether a mail server has been configured.</li> <li>• <b>apic cloud-settings:topology</b> - Returns summary details of the configuration of your cloud topology.</li> <li>• <b>apic cloud-settings:update</b> - Update the Cloud Setting.</li> </ul>
<b>apic configured-gateway-services</b>	Commands for Configured Gateway Services in a Catalog or Space.	<ul style="list-style-type: none"> <li>• <b>apic configured-gateway-services</b> - Configured Gateway Services operations.</li> <li>• <b>apic configured-gateway-services:clear</b> - Delete all Configured Gateway Services from a Catalog or Space.</li> <li>• <b>apic configured-gateway-services:create</b> - Configure a Gateway Service in a Catalog or Space.</li> <li>• <b>apic configured-gateway-services:delete</b> - Delete a Configured Gateway Service from a Catalog or Space.</li> <li>• <b>apic configured-gateway-services:get</b> - Obtain the details of a Configured Gateway Service in a Catalog or Space.</li> <li>• <b>apic configured-gateway-services:list</b> - List the Configured Gateway Services in a Catalog or Space.</li> </ul>
<b>apic credentials</b>	Credential commands	<ul style="list-style-type: none"> <li>• <b>apic credentials</b> - Application Credential collection operations.</li> <li>• <b>apic credentials:clear</b> - Clear the Application Credentials.</li> <li>• <b>apic credentials:create</b> - Create a Application Credential.</li> <li>• <b>apic credentials:delete</b> - Delete the Application Credential by name or ID.</li> <li>• <b>apic credentials:get</b> - Get the Application Credential by name or ID.</li> <li>• <b>apic credentials:list</b> - List the Application Credentials.</li> <li>• <b>apic credentials:reset</b> - Reset the client id and client secret.</li> <li>• <b>apic credentials:reset-client-secret</b> - Reset the client secret.</li> <li>• <b>apic credentials:update</b> - Update the Application Credential by name or ID.</li> <li>• <b>apic credentials:verify-client-secret</b> - Verify the client secret.</li> </ul>
<b>apic extensions</b>	Extension commands	<ul style="list-style-type: none"> <li>• <b>apic extensions</b> - Extension collection operations.</li> <li>• <b>apic extensions:document</b> - Get the document for an extension by id, or by name and version.</li> <li>• <b>apic extensions:get</b> - Get an extension by id, or by name and version.</li> <li>• <b>apic extensions:list</b> - List all versions of an extension given the name.</li> <li>• <b>apic extensions:list-all</b> - List all versions of all extensions.</li> </ul>
<b>apic gateway-services</b>	Gateway service commands	<ul style="list-style-type: none"> <li>• <b>apic gateway-services</b> - Gateway Service collection operations.</li> <li>• <b>apic gateway-services:clear</b> - Clear the Gateway Services.</li> <li>• <b>apic gateway-services:create</b> - Create a Gateway Service.</li> <li>• <b>apic gateway-services:delete</b> - Delete the Gateway Service by name or ID.</li> <li>• <b>apic gateway-services:get</b> - Get the Gateway Service by name or ID.</li> <li>• <b>apic gateway-services:list</b> - List the Gateway Services.</li> <li>• <b>apic gateway-services:reset-oauth-secret</b> - Reset the OAuth cryptographic</li> <li>• <b>apic gateway-services:update</b> - Update the Gateway Service by name or ID.</li> </ul>



Command	Description	Sub-commands
<b>apic integrations</b>	Integrations commands	<ul style="list-style-type: none"> <li>• <b>apic integrations</b> - Integration collection operations.</li> <li>• <b>apic integrations:clear</b> - Clear the Integrations.</li> <li>• <b>apic integrations:clear-all</b> - Clear all of the Integrations in all of the collections.</li> <li>• <b>apic integrations:create</b> - Create a Integration.</li> <li>• <b>apic integrations:delete</b> - Delete the Integration by name or ID.</li> <li>• <b>apic integrations:get</b> - Get the Integration by name or ID.</li> <li>• <b>apic integrations:list</b> - List the Integrations.</li> <li>• <b>apic integrations:list-all</b> - List all Integrations in all collections.</li> <li>• <b>apic integrations:update</b> - Update the Integration by name or ID.</li> </ul>
<b>apic keystores</b>	Keystores commands	<ul style="list-style-type: none"> <li>• <b>apic keystores</b> - Keystore collection operations.</li> <li>• <b>apic keystores:clear</b> - Clear the Keystores.</li> <li>• <b>apic keystores:create</b> - Create a Keystore.</li> <li>• <b>apic keystores:delete</b> - Delete the Keystore by name or ID.</li> <li>• <b>apic keystores:get</b> - Get the Keystore by name or ID.</li> <li>• <b>apic keystores:list</b> - List the Keystores.</li> <li>• <b>apic keystores:update</b> - Update the Keystore by name or ID.</li> </ul>
<b>apic mail-servers</b>	Mail server commands	<ul style="list-style-type: none"> <li>• <b>apic mail-servers</b> - Mail Server collection operations.</li> <li>• <b>apic mail-servers:clear</b> - Clear the Mail Servers.</li> <li>• <b>apic mail-servers:create</b> - Create a Mail Server.</li> <li>• <b>apic mail-servers:delete</b> - Delete the Mail Server by name or ID.</li> <li>• <b>apic mail-servers:get</b> - Get the Mail Server by name or ID.</li> <li>• <b>apic mail-servers:list</b> - List the Mail Servers.</li> <li>• <b>apic mail-servers:test-connection</b> - Test a mail server connection.</li> <li>• <b>apic mail-servers:update</b> - Update the Mail Server by name or ID.</li> </ul>
<b>apic notification-templates</b>	Notification template commands	<ul style="list-style-type: none"> <li>• <b>apic notification-templates</b> - Notification Template collection operations.</li> <li>• <b>apic notification-templates:get</b> - Get the Notification Template by name or ID.</li> <li>• <b>apic notification-templates:list</b> - List the Notification Templates.</li> <li>• <b>apic notification-templates:list-all</b> - List all Notification Templates in all collections.</li> <li>• <b>apic notification-templates:update</b> - Update the Notification Template by name or ID.</li> </ul>
<b>apic permissions</b>	Permissions commands	<ul style="list-style-type: none"> <li>• <b>apic permissions</b> - Permission collection operations.</li> <li>• <b>apic permissions:get</b> - Get the Permission by name or ID.</li> <li>• <b>apic permissions:list</b> - List the Permissions.</li> <li>• <b>apic permissions:list-all</b> - List all Permissions in all collections.</li> </ul>
<b>apic portal-services</b>	Portal services commands	<ul style="list-style-type: none"> <li>• <b>apic portal-services</b> - Portal Service collection operations.</li> <li>• <b>apic portal-services:clear</b> - Clear the Portal Services.</li> <li>• <b>apic portal-services:create</b> - Create a Portal Service.</li> <li>• <b>apic portal-services:delete</b> - Delete the Portal Service by name or ID.</li> <li>• <b>apic portal-services:get</b> - Get the Portal Service by name or ID.</li> <li>• <b>apic portal-services:list</b> - List the Portal Services.</li> <li>• <b>apic portal-services:update</b> - Update the Portal Service by name or ID.</li> </ul>
<b>apic services</b>	Services commands	<ul style="list-style-type: none"> <li>• <b>apic services</b> - Service collection operations.</li> <li>• <b>apic services:clear</b> - Clear the Services.</li> <li>• <b>apic services:clear-all</b> - Clear all Services in all collections.</li> <li>• <b>apic services:create</b> - Create a Service.</li> <li>• <b>apic services:delete</b> - Delete a Service.</li> <li>• <b>apic services:get</b> - Get the Service by name and version.</li> <li>• <b>apic services:list</b> - List the Services.</li> <li>• <b>apic services:list-all</b> - List all Services in all collections.</li> <li>• <b>apic services:update</b> - Update the Service by name and version.</li> </ul>
<b>apic space-settings</b>	Space settings commands	<ul style="list-style-type: none"> <li>• <b>apic space-settings:get</b> - Get the Space Setting.</li> <li>• <b>apic space-settings:update</b> - Update the Space Setting.</li> </ul>
<b>apic spaces</b>	Space commands	<ul style="list-style-type: none"> <li>• <b>apic spaces</b> - Space collection operations.</li> <li>• <b>apic spaces:clear</b> - Clear the Spaces.</li> <li>• <b>apic spaces:create</b> - Create a Space.</li> <li>• <b>apic spaces:delete</b> - Delete the Space by name or ID.</li> <li>• <b>apic spaces:get</b> - Get the Space by name or ID.</li> <li>• <b>apic spaces:list</b> - List the Spaces.</li> <li>• <b>apic spaces:transfer-owner</b> - Transfer owner to a new member.</li> <li>• <b>apic spaces:update</b> - Update the Space by name or ID.</li> </ul>

Command	Description	Sub-commands
<code>apic tls-client-profiles</code>	TLS client profile commands	<ul style="list-style-type: none"> <li>• <code>apic tls-client-profiles</code> - TLS Client Profile collection operations.</li> <li>• <code>apic tls-client-profiles:clear</code> - Clear the TLS Client Profiles.</li> <li>• <code>apic tls-client-profiles:clear-all</code> - Clear all of the TLS Client Profiles in the collection.</li> <li>• <code>apic tls-client-profiles:create</code> - Create a TLS Client Profile.</li> <li>• <code>apic tls-client-profiles:delete</code> - Delete the TLS Client Profile by name or ID.</li> <li>• <code>apic tls-client-profiles:get</code> - Get the TLS Client Profile by name or ID.</li> <li>• <code>apic tls-client-profiles:list</code> - List the TLS Client Profiles.</li> <li>• <code>apic tls-client-profiles:list-all</code> - Lists all of the TLS Client Profiles in a collection.</li> <li>• <code>apic tls-client-profiles:update</code> - Update the TLS Client Profile by name or ID.</li> </ul>
<code>apic tls-server-profiles</code>	TLS server profile commands	<ul style="list-style-type: none"> <li>• <code>apic tls-server-profiles</code> - TLS Server Profile collection operations.</li> <li>• <code>apic tls-server-profiles:clear</code> - Clear the TLS Server Profiles.</li> <li>• <code>apic tls-server-profiles:clear-all</code> - Clear all of the TLS Server Profiles in the collection.</li> <li>• <code>apic tls-server-profiles:create</code> - Create a TLS Server Profile.</li> <li>• <code>apic tls-server-profiles:delete</code> - Delete the TLS Server Profile by name or ID.</li> <li>• <code>apic tls-server-profiles:get</code> - Get the TLS Server Profile by name or ID.</li> <li>• <code>apic tls-server-profiles:list</code> - List the TLS Server Profiles.</li> <li>• <code>apic tls-server-profiles:list-all</code> - List all of the TLS Server Profiles in the collection.</li> <li>• <code>apic tls-server-profiles:update</code> - Update the TLS Server Profile by name or ID.</li> </ul>
<code>apic truststores</code>	Truststore commands	<ul style="list-style-type: none"> <li>• <code>apic truststores</code> - Truststore collection operations.</li> <li>• <code>apic truststores:clear</code> - Clear the Truststores.</li> <li>• <code>apic truststores:create</code> - Create a Truststore.</li> <li>• <code>apic truststores:delete</code> - Delete the Truststore by name or ID.</li> <li>• <code>apic truststores:get</code> - Get the Truststore by name or ID.</li> <li>• <code>apic truststores:list</code> - List the Truststores.</li> <li>• <code>apic truststores:update</code> - Update the Truststore by name or ID.</li> </ul>
<code>apic user-registries</code>	User registry commands	<ul style="list-style-type: none"> <li>• <code>apic user-registries</code> - User Registry collection operations.</li> <li>• <code>apic user-registries:clear</code> - Clear the User Registries.</li> <li>• <code>apic user-registries:create</code> - Create a User Registry.</li> <li>• <code>apic user-registries:delete</code> - Delete the User Registry by name or ID.</li> <li>• <code>apic user-registries:execute</code> - Execute a User Registry operation.</li> <li>• <code>apic user-registries:get</code> - Get the User Registry by name or ID.</li> <li>• <code>apic user-registries:list</code> - List the User Registries.</li> <li>• <code>apic user-registries:search</code> - search for users in a user registry.</li> <li>• <code>apic user-registries:test-connection</code> - Test a User Registry connection.</li> <li>• <code>apic user-registries:update</code> - Update the User Registry by name or ID.</li> </ul>
<code>apic user-registry-settings</code>	User registry settings commands	<ul style="list-style-type: none"> <li>• <code>apic user-registry-settings:get</code> - Get the User Registry Setting.</li> <li>• <code>apic user-registry-settings:update</code> - Update the User Registry Setting.</li> </ul>
<code>apic users</code>	Users commands	<ul style="list-style-type: none"> <li>• <code>apic users</code> - User collection operations.</li> <li>• <code>apic users:clear</code> - Clear the Users.</li> <li>• <code>apic users:create</code> - Create a User.</li> <li>• <code>apic users:delete</code> - Delete the User by name or ID.</li> <li>• <code>apic users:get</code> - Get the User by name or ID.</li> <li>• <code>apic users:list</code> - List the Users.</li> <li>• <code>apic users:request-password-reset</code> - send a password reset link.</li> <li>• <code>apic users:search-provider</code> - Search for provider users from an organization.</li> <li>• <code>apic users:update</code> - Update the User by name or ID.</li> </ul>
<code>apic webhooks</code>	Webhooks commands	<ul style="list-style-type: none"> <li>• <code>apic webhooks</code> - Webhooks collection operations.</li> <li>• <code>apic webhooks:get</code> - Get the Webhooks by name or ID.</li> <li>• <code>apic webhooks:list</code> - List the Webhooks.</li> <li>• <code>apic webhooks:update</code> - Update the Webhooks by name or ID.</li> </ul>

## API development and management commands

A summary of the core commands in the IBM® API Connect developer toolkit.

### Authenticating

Use the `apic login` command to authenticate to an API Manager service, and the `apic logout` command to remove your local authentication credentials.

Note: When you authenticate successfully, your credentials are stored, in plain text, in the file `Linux .netrc` or `Windows _netrc`. Therefore, you should set the file permissions in such a way that your credentials are not accessible by others.

For full details on how to log in to your management server from the CLI, see [Logging in to the management server](#).

## Configuring the command-line tool to use TLS certificates

API Manager uses TLS profiles to secure data transmission. For information on how to create a TLS profile in API Manager, see [TLS profiles](#).

To configure the toolkit command-line tool to use certificates to communicate with an API Manager that has TLS profiles that are enabled, follow these steps:

For more information about the `NODE_EXTRA_CA_CERTS` environment variable, see [Node.js documentation](#).

## Creating and managing local files

You create and work with API and Product definition YAML files locally before you stage them to API Manager.

To create a local API definition file, use the `apic create:api` command. To create a local Product definition file, use the `apic create:product` command.

Use the `apic apis` and `apic products` commands to list API Manager artifacts of the specified type.

To validate the syntactical correctness of a local API or Product definition file, use the `apic validate` command.

To create a draft API in API Manager from a local API or Product definition file, use the `apic draft-apis:create` and `apic draft-products:create` commands, respectively.

Note: If the OpenAPI file that defines your API uses a `$ref` field to reference a fragment of OpenAPI code that is defined in a separate file, the `$ref` field is replaced with the contents of the target file before an API is validated, created in draft, staged, or published. For more information, see [Using \\$ref to reuse code fragments in your OpenAPI files](#).

## Working with Catalogs and Spaces

To create a Catalog, use the `apic catalog:create` command. To view information on a Catalog, use the `apic catalog:get` command; to list all Catalogs that are contained in organizations that the currently authenticated user is a member of, use the `apic catalogs` command.

You can use a *Space* to partition a Catalog so multiple teams can manage Products and APIs independently in a single Catalog. A Space is conceptually like a subcatalog, except that Products and APIs in all Spaces in a given Catalog are published to the same developer portal. For more information about Spaces, see [Using syndication in IBM API Connect](#).

To enable Spaces for a Catalog, use the following command:

```
apic catalogs:set catalog_name --spaces enabled
```

Use the toolkit `apic spaces` commands to create and manage Spaces:

- `apic spaces` - List Spaces contained in a Catalog.
- `apic:spaces create` - Create a Space in a Catalog.
- `apic:spaces get` - Get information on a Space in a Catalog.
- `apic:spaces set` - Set information on a Space in a Catalog.
- `apic:spaces delete` - Delete a Space in a Catalog.

## Command summary

The following tables summarize `apic` commands for API development and management.

Table 1. Summary of general-purpose commands

Command	Description	Subcommands
<code>apic config</code>	List and manage configuration variables. For more information, see <a href="#">Using configuration variables</a> . With no subcommand, lists values of defined configuration variables.	<ul style="list-style-type: none"><li>• <code>apic config</code> - Manage configuration variables</li><li>• <code>apic config:clear</code> - Delete all configuration variables</li><li>• <code>apic config:delete</code> - Delete a configuration variable</li><li>• <code>apic config:get</code> - Get a configuration variable</li><li>• <code>apic config:list</code> - List the application and global configuration variables</li><li>• <code>apic config:set</code> - Set or update configuration variables</li></ul>
<code>apic create</code>	Create a draft API or Product definition YAML file.	<ul style="list-style-type: none"><li>• <code>create:api</code> - Create a draft API OpenAPI definition YAML file</li><li>• <code>create:product</code> - Create a Product definition YAML file</li></ul>

Command	Description	Subcommands
<code>apic extensions</code>	Manage OpenAPI extensions in a Catalog. With no subcommand, lists the extensions in the production Catalog.	<ul style="list-style-type: none"> <li><code>apic extensions</code> - Extension collection operations.</li> <li><code>apic extensions:clear</code> - Delete all versions of an extension given the name.</li> <li><code>apic extensions:clear-all</code> - Delete all versions of all extensions.</li> <li><code>apic extensions:clone</code> - Download all versions of all extensions to your local drive.</li> <li><code>apic extensions:create</code> - Create an extension.</li> <li><code>apic extensions:delete</code> - Delete an extension by ID, or by name and version.</li> <li><code>apic extensions:document</code> - Get the document for an extension by ID, or by name and version.</li> <li><code>apic extensions:get</code> - Get an extension by ID, or by name and version.</li> <li><code>apic extensions:list</code> - List all versions of an extension given the name.</li> <li><code>apic extensions:list-all</code> - List all versions of all extensions.</li> <li><code>apic extensions:update</code> - Update an extension by ID, or by name and version.</li> </ul>
<code>apic lb4</code>	<p>Create LoopBack® project and project artifacts. With no subcommand, creates a new LoopBack project.</p> <p>Note: To use the <code>apic lb4</code> command, you need to add a double dash, <code>--</code>, after the <code>lb4</code> subcommand to tell <code>apic</code> that the options behind the <code>--</code> are for the subcommand belonging to <code>lb4</code>. The second <code>--</code> is to signify the end of the options. For example,</p> <pre>\$ apic lb4 controller -- --help</pre> <p>Note: You can use</p> <pre>\$ apic lb4 --help</pre> <p>to show a list of commands that are available, and you can use</p> <pre>\$ apic lb4 subcommand -- --help</pre> <p>to show the usage of a subcommand.</p>	<ul style="list-style-type: none"> <li><code>app</code> - Create a new LoopBack project (default)</li> <li><code>controller</code> - Add a controller to a LoopBack 4 application</li> <li><code>datasource</code> - Add a datasource to a LoopBack 4 application</li> <li><code>discover</code> - Discover models from relational databases</li> <li><code>example</code> - Download one of LoopBack 4 example projects</li> <li><code>extension</code> - Create a LoopBack 4 extension</li> <li><code>import-lb3-models</code> - Import one or more LoopBack 3 models to a LoopBack 4 application</li> <li><code>interceptor</code> - Generate interceptors</li> <li><code>model</code> - Add a new model to a LoopBack 4 application</li> <li><code>observer</code> - Generate life cycle observers for application start and stop</li> <li><code>openapi</code> - Generate controllers and models from OpenAPI specs</li> <li><code>relation</code> - Add a relation between two models in a LoopBack 4 application</li> <li><code>repository</code> - Add a new repository for a selected model to a LoopBack 4 application</li> <li><code>rest-crud</code> - Generate rest configs for model endpoints</li> <li><code>service</code> - Add a new remote or local service to a LoopBack 4 application</li> <li><code>update</code> - Check or update project dependencies of LoopBack modules</li> </ul>
<code>apic login</code>	Log in to API Manager.	<p>None. Specify server and credentials with the required flags:</p> <ul style="list-style-type: none"> <li><code>-p, --password password</code></li> <li><code>-r, --realm realm</code>. Determines the identity provider that is used to authenticated the supplied user ID, and the scope of the tasks that can be performed after successful log in.</li> <li><code>-s, --server mgmt_endpoint_url</code>. Use the value of the platform API endpoint URL for this entry.</li> <li><code>-u, --username user_name</code></li> </ul> <p>For full details on how to log in to your management server from the CLI, see <a href="#">Logging in to the management server</a>.</p>
<code>apic logout</code>	Log out from API Manager.	<p>None. Specify server with the required flag:</p> <ul style="list-style-type: none"> <li><code>-s, --server mgmt_service</code>.</li> </ul> <p>You can append the port number to the server name if it is not the default value of 443.</p>
<code>apic validate</code>	Validate API or Product definition YAML file.	None

Table 2. Summary of commands to manage APIs, Products, and Catalogs

Command	Description	Subcommands
<code>apic apis</code>	List and manage APIs that are staged or published to Catalog or Space. Default subcommand is <code>list-all</code> .	<ul style="list-style-type: none"> <li><code>apic apis</code> - API collection operations</li> <li><code>apic apis:get</code> - Get the OpenAPI definition YAML file for an API by name and version, or by ID</li> <li><code>apic apis:list</code> - List an API by name</li> <li><code>apic apis:list-all</code> - List all APIs in a Catalog or Space</li> <li><code>apic apis:update</code> - Update an API by name</li> </ul>

Command	Description	Subcommands
<b>apic apps</b>	List and manage developer applications that are registered in a consumer organization. Default subcommand is <b>list</b> .	<ul style="list-style-type: none"> <li>• <b>apic apps</b> - Application collection operations</li> <li>• <b>apic apps:clear</b> - Clear all applications</li> <li>• <b>apic apps:create</b> - Create an application</li> <li>• <b>apic apps:delete</b> - Delete an application by name or ID</li> <li>• <b>apic apps:get</b> - Get an application object by name</li> <li>• <b>apic apps:list</b> - List all the applications in a consumer organization</li> <li>• <b>apic apps:update</b> - Update an application by name or ID</li> </ul>
<b>apic catalogs</b>	List and manage Catalogs in a provider organization. Default subcommand is <b>list</b> .	<ul style="list-style-type: none"> <li>• <b>apic catalogs:clear</b> - Clear all Catalogs from provider organization</li> <li>• <b>apic catalogs:create</b> - Create a Catalog</li> <li>• <b>apic catalogs:delete</b> - Delete a Catalog by name or ID</li> <li>• <b>apic catalogs:get</b> - Get a Catalog object by name or ID</li> <li>• <b>apic catalogs:list</b> - List all the Catalogs in a provider organization</li> <li>• <b>apic catalogs:transfer-owner</b> - Transfer the ownership of a Catalog to another user</li> <li>• <b>apic catalogs:update</b> - Update a Catalog by name or ID</li> </ul>
<b>apic catalog-settings</b>	Manage the configuration settings for a Catalog	<ul style="list-style-type: none"> <li>• <b>apic catalog-settings:get</b> - Get the Catalog settings.</li> <li>• <b>apic catalog-settings:update</b> - Update the Catalog settings.</li> </ul>
<b>apic drafts</b>	List and manage the draft API and Product definitions in a provider organization. Default subcommand is <b>list</b> .	<ul style="list-style-type: none"> <li>• <b>apic drafts</b> - Draft collection operations</li> <li>• <b>apic drafts:clear</b> - Delete all draft APIs and Products in a provider organization</li> <li>• <b>apic drafts:list</b> - List the draft APIs and Products in a provider organization</li> </ul>
<b>apic draft-apis</b>	List and manage the draft API definitions in a provider organization. Default subcommand is <b>list-all</b> .	<ul style="list-style-type: none"> <li>• <b>apic draft-apis</b> - Draft API collection operations</li> <li>• <b>apic draft-apis:clear</b> - Delete all versions of a draft API by name</li> <li>• <b>apic draft-apis:clear-all</b> - Delete all draft APIs in a provider organization</li> <li>• <b>apic draft-apis:clone</b> - Clone all draft APIs in a provider organization</li> <li>• <b>apic draft-apis:create</b> - Create a draft API</li> <li>• <b>apic draft-apis:delete</b> - Delete a Draft API by name and version</li> <li>• <b>apic draft-apis:document</b> - Get the Draft API document by name and version</li> <li>• <b>apic draft-apis:get</b> - Get the OpenAPI definition YAML file for an API by name and version</li> <li>• <b>apic draft-apis:list</b> - List all versions of a draft API by name</li> <li>• <b>apic draft-apis:list-all</b> - List all draft APIs in a provider organization</li> <li>• <b>apic draft-apis:update</b> - Update a draft API by name and version, supplying the revised OpenAPI definition file</li> <li>• <b>apic draft-apis:update-wsdl</b> - Update a draft SOAP API by name and version, supplying the revised WSDL definition file in a .wsdl or .zip format Note: There must be a match of the service names in the existing WSDL with the revised WSDL. If a match is found, the WSDL service is converted into a YAML file that is used to update the API. If no match is found, an error message is displayed.</li> <li>• <b>apic draft-apis:validate</b> - Validates a draft API</li> <li>• <b>apic draft-apis:wsdl</b> - Get the Draft API WSDL document by name and version</li> </ul>
<b>apic draft-products</b>	List and manage the draft Product definitions in a provider organization. Default subcommand is <b>list-all</b> .	<ul style="list-style-type: none"> <li>• <b>apic draft-products</b> - Draft Product collection operations</li> <li>• <b>apic draft-products:clear</b> - Delete all versions of a draft Product by name</li> <li>• <b>apic draft-products:clear-all</b> - Delete all draft Products in a provider organization</li> <li>• <b>apic draft-products:create</b> - Create a draft Product</li> <li>• <b>apic draft-products:delete</b> - Delete a draft Product by name and version</li> <li>• <b>apic draft-products:get</b> - Get the Draft Product by name and version</li> <li>• <b>apic draft-products:list</b> - List the Draft Products</li> <li>• <b>apic draft-products:list-all</b> - List all draft Products in a provider organization</li> <li>• <b>apic draft-products:publish-all</b> - Publishes the Draft Product</li> <li>• <b>apic draft-products:update</b> - Update a draft Product by name and version, supplying the revised definition file</li> <li>• <b>apic draft-products:validate</b> - Validates a draft Product</li> </ul>
<b>apic members</b>	List and manage the members of a provider organization, consumer organization, Catalog, or Space. Default subcommand is <b>list</b> .	<ul style="list-style-type: none"> <li>• <b>apic members</b> - Member operations</li> <li>• <b>apic members:clear</b> - Delete all members</li> <li>• <b>apic members:create</b> - Create a member</li> <li>• <b>apic members:delete</b> - Delete a member by name or ID</li> <li>• <b>apic members:get</b> - Get a member object by name or ID</li> <li>• <b>apic members:list</b> - List all members</li> <li>• <b>apic members:update</b> - Update the member by name or ID</li> </ul>
<b>apic member-invitations</b>	List and manage member invitations. A member invitation is created when a user is invited to be a member of a provider organization, consumer organization, Catalog, or Space. Default subcommand is <b>list</b> .	<ul style="list-style-type: none"> <li>• <b>apic member-invitations</b> - Member invitation collection operations.</li> <li>• <b>apic member-invitations:clear</b> - Delete all member invitations.</li> <li>• <b>apic member-invitations:create</b> - Create a member invitation.</li> <li>• <b>apic member-invitations:delete</b> - Delete a member invitation by name or ID.</li> <li>• <b>apic member-invitations:get</b> - Get a member invitation object by name or ID.</li> <li>• <b>apic member-invitations:list</b> - List all member invitations.</li> <li>• <b>apic member-invitations:update</b> - Update a member invitation by name or ID.</li> </ul>

Command	Description	Subcommands
<b>apic orgs</b>	List and manage provider organizations, and the admin organization. Default subcommand is <b>list</b> .	<ul style="list-style-type: none"> <li>• <b>apic orgs</b> - Organization collection operations</li> <li>• <b>apic orgs:clear</b> - Delete all organizations</li> <li>• <b>apic orgs:create</b> - Create an organization</li> <li>• <b>apic orgs:delete</b> - Delete an organization by name or ID</li> <li>• <b>apic orgs:get</b> - Get an organization object by name or ID</li> <li>• <b>apic orgs:list</b> - List all organizations</li> <li>• <b>apic orgs:transfer-owner</b> - Transfer ownership of an organization</li> <li>• <b>apic orgs:update</b> - Update an organization by name or ID</li> </ul>
<b>apic org-settings</b>	Manage settings for provider organizations, and the admin organization.	<ul style="list-style-type: none"> <li>• <b>apic org-settings:get</b> - Get the settings object for an organization</li> <li>• <b>apic org-settings:update</b> - Update the settings for an organization</li> </ul>
<b>apic policies</b>	List and manage policies in a Catalog. Default subcommand is <b>list</b> .	<ul style="list-style-type: none"> <li>• <b>apic policies</b> - Policy collection operations</li> <li>• <b>apic policies:clear</b> - Clear the Policies</li> <li>• <b>apic policies:clear-all</b> - Clear all Policies in all collections</li> <li>• <b>apic policies:create</b> - Create a Policy</li> <li>• <b>apic policies:delete</b> - Delete a Policy</li> <li>• <b>apic policies:get</b> - Get the Policy by name and version</li> <li>• <b>apic policies:list</b> - List the Policies</li> <li>• <b>apic policies:list-all</b> - List all Policies in all collections</li> <li>• <b>apic policies:update</b> - Update the Policy by name and version</li> </ul>
<b>apic products</b>	List and manage Products that are staged or published to Catalog or Space. Default subcommand is <b>list-all</b> .	<ul style="list-style-type: none"> <li>• <b>apic products</b> - Product collection operations</li> <li>• <b>apic products:clear</b> - Delete all versions of a Product by name</li> <li>• <b>apic products:clear-all</b> - Delete all Products</li> <li>• <b>apic products:delete</b> - Delete a Product by name and version</li> <li>• <b>apic products:execute-migration-target</b> - Products execute migration target operations</li> <li>• <b>apic products:get</b> - Get a Product object by name and version</li> <li>• <b>apic products:list</b> - List all versions of a Product by name</li> <li>• <b>apic products:list-all</b> - List all Products</li> <li>• <b>apic products:publish</b> - Publish a Product to a Catalog or Space, supplying the Product definition YAML file</li> <li>• <b>apic products:replace</b> - Replace a Product with another Product</li> <li>• <b>apic products:set-migration-target</b> - Set the target Product for migrating subscriptions from a Product</li> <li>• <b>apic products:supersede</b> - Supersede a Product with another Product</li> <li>• <b>apic products:update</b> - Update a Product by name and version, supplying the revised Product definition YAML file</li> </ul>
<b>apic identity-providers</b>	View information about identity providers.	<ul style="list-style-type: none"> <li>• <b>apic identity-providers</b> - Identity provider operations</li> <li>• <b>apic identity-providers:list</b> - List the identity providers.</li> </ul>
<b>apic spaces</b>	List and manage Spaces contained in a Catalog. Default subcommand is <b>list</b> .	<ul style="list-style-type: none"> <li>• <b>apic spaces</b> - Space collection operations</li> <li>• <b>apic spaces:clear</b> - Delete all Spaces from a Catalog</li> <li>• <b>apic spaces:create</b> - Create a Space in a Catalog</li> <li>• <b>apic spaces:delete</b> - Delete a Space from a Catalog, by name or ID</li> <li>• <b>apic spaces:get</b> - Get a Space object by name or ID</li> <li>• <b>apic spaces:list</b> - List all the Spaces in a Catalog</li> <li>• <b>apic spaces:transfer-owner</b> - Transfer the ownership of a Space to another user</li> <li>• <b>apic spaces:update</b> - Update a Space by name or ID</li> </ul>
<b>apic subscriptions</b>	List and manage subscriptions in a Product or a Catalog. Default subcommand is <b>list</b> .	<ul style="list-style-type: none"> <li>• <b>apic subscriptions</b> - Subscription collection operations</li> <li>• <b>apic subscriptions:clear</b> - Clear the Subscriptions</li> <li>• <b>apic subscriptions:create</b> - Create a Subscription</li> <li>• <b>apic subscriptions:delete</b> - Delete the Subscription by name or ID</li> <li>• <b>apic subscriptions:get</b> - Get the Subscription by name or ID</li> <li>• <b>apic subscriptions:list</b> - List the Subscriptions</li> <li>• <b>apic subscriptions:update</b> - Update the Subscription by name or ID</li> </ul>

Table 3. Summary of other commands for API developers and managers

Command	Description	Subcommands
<b>apic consumer-orgs</b>	Manage consumer organizations	<ul style="list-style-type: none"> <li>• <b>apic consumer-orgs</b> - Consumer Organization collection operations.</li> <li>• <b>apic consumer-orgs:clear</b> - Clear the Consumer Organizations.</li> <li>• <b>apic consumer-orgs:create</b> - Create a Consumer Organization.</li> <li>• <b>apic consumer-orgs:delete</b> - Delete the Consumer Organization by name or ID.</li> <li>• <b>apic consumer-orgs:get</b> - Get the Consumer Organization by name or ID.</li> <li>• <b>apic consumer-orgs:list</b> - List the Consumer Organizations.</li> <li>• <b>apic consumer-orgs:transfer-owner</b> - Transfer owner to a new member.</li> <li>• <b>apic consumer-orgs:update</b> - Update the Consumer Organization by name or ID.</li> </ul>

Command	Description	Subcommands
<code>apic consumer-org-settings</code>	Manage consumer organization settings	<ul style="list-style-type: none"> <li>• <code>apic consumer-org-settings:delete</code> - Delete the Consumer Organization Setting.</li> <li>• <code>apic consumer-org-settings:get</code> - Get the Consumer Organization Setting.</li> <li>• <code>apic consumer-org-settings:update</code> - Update the Consumer Organization Setting.</li> </ul>
<code>apic groups</code>	Manage groups	<ul style="list-style-type: none"> <li>• <code>apic groups</code> - Group collection operations.</li> <li>• <code>apic groups:clear</code> - Clear the Groups.</li> <li>• <code>apic groups:create</code> - Create a Group.</li> <li>• <code>apic groups:delete</code> - Delete the Group by name or ID.</li> <li>• <code>apic groups:get</code> - Get the Group by name or ID.</li> <li>• <code>apic groups:list</code> - List the Groups.</li> <li>• <code>apic groups:update</code> - Update the Group by name or ID.</li> </ul>
<code>apic invitations</code>	Manage invitations. An invitation is created when a user is invited to be the owner of a provider organization, consumer organization, Catalog, or Space.	<ul style="list-style-type: none"> <li>• <code>apic invitations</code> - Invitation collection operations.</li> <li>• <code>apic invitations:clear</code> - Delete all invitations.</li> <li>• <code>apic invitations:create</code> - Create an invitation.</li> <li>• <code>apic invitations:delete</code> - Delete an invitation by name or ID.</li> <li>• <code>apic invitations:get</code> - Get the details of an invitation by name or ID.</li> <li>• <code>apic invitations:list</code> - List all invitations.</li> <li>• <code>apic invitations:update</code> - Update an invitation by name or ID.</li> </ul>
<code>apic member-invitations</code>	Manage member invitations. A member invitation is created when a user is invited to be a member of a provider organization, consumer organization, Catalog, or Space.	<ul style="list-style-type: none"> <li>• <code>apic member-invitations</code> - Member invitation collection operations.</li> <li>• <code>apic member-invitations:clear</code> - Delete all member invitations.</li> <li>• <code>apic member-invitations:create</code> - Create a member invitation.</li> <li>• <code>apic member-invitations:delete</code> - Delete a member invitation by name or ID.</li> <li>• <code>apic member-invitations:get</code> - Get the details of a member invitation by name or ID.</li> <li>• <code>apic member-invitations:list</code> - List all member invitations.</li> <li>• <code>apic member-invitations:update</code> - Update a member invitation by name or ID.</li> </ul>
<code>apic members</code>		<ul style="list-style-type: none"> <li>• <code>apic members</code> - Member collection operations.</li> <li>• <code>apic members:clear</code> - Clear the Members.</li> <li>• <code>apic members:create</code> - Create a Member.</li> <li>• <code>apic members:delete</code> - Delete the Member by name or ID.</li> <li>• <code>apic members:get</code> - Get the Member by name or ID.</li> <li>• <code>apic members:list</code> - List the Members.</li> <li>• <code>apic members:update</code> - Update the Member by name or ID.</li> </ul>
<code>apic registrations</code>		<ul style="list-style-type: none"> <li>• <code>apic registrations</code> - Registration collection operations.</li> <li>• <code>apic registrations:clear</code> - Clear the Registrations.</li> <li>• <code>apic registrations:create</code> - Create a Registration.</li> <li>• <code>apic registrations:delete</code> - Delete the Registration by name or ID.</li> <li>• <code>apic registrations:get</code> - Get the Registration by name or ID.</li> <li>• <code>apic registrations:list</code> - List the Registrations.</li> <li>• <code>apic registrations:update</code> - Update the Registration by name or ID.</li> </ul>

Command	Description	Subcommands
apic role-defaults		<ul style="list-style-type: none"> <li>• <b>apic role-defaults</b> - Role Default collection operations.</li> <li>• <b>apic role-defaults:clear</b> - Clear the Role Defaults.</li> <li>• <b>apic role-defaults:create</b> - Create a Role Default.</li> <li>• <b>apic role-defaults:delete</b> - Delete the Role Default by name or ID.</li> <li>• <b>apic role-defaults:get</b> - Get the Role Default by name or ID.</li> <li>• <b>apic role-defaults:list</b> - List the Role Defaults.</li> <li>• <b>apic role-defaults:list-all</b> - List all Role Defaults in all collections.</li> <li>• <b>apic role-defaults:update</b> - Update the Role Default by name or ID.</li> </ul>
apic roles		<ul style="list-style-type: none"> <li>• <b>apic roles</b> - Role collection operations.</li> <li>• <b>apic roles:clear</b> - Clear the Roles.</li> <li>• <b>apic roles:create</b> - Create a Role.</li> <li>• <b>apic roles:delete</b> - Delete the Role by name or ID.</li> <li>• <b>apic roles:get</b> - Get the Role by name or ID.</li> <li>• <b>apic roles:list</b> - List the Roles.</li> <li>• <b>apic roles:update</b> - Update the Role by name or ID.</li> </ul>
apic tasks		<ul style="list-style-type: none"> <li>• <b>apic tasks</b> - Task collection operations.</li> <li>• <b>apic tasks:clear</b> - Clear the Tasks.</li> <li>• <b>apic tasks:create</b> - Create a Task.</li> <li>• <b>apic tasks:delete</b> - Delete the Task by name or ID.</li> <li>• <b>apic tasks:get</b> - Get the Task by name or ID.</li> <li>• <b>apic tasks:list</b> - List the Tasks.</li> <li>• <b>apic tasks:update</b> - Update the Task by name or ID.</li> </ul>
apic user-registries		<ul style="list-style-type: none"> <li>• <b>apic user-registries</b> - User Registry collection operations.</li> <li>• <b>apic user-registries:clear</b> - Clear the User Registries.</li> <li>• <b>apic user-registries:create</b> - Create a User Registry.</li> <li>• <b>apic user-registries:delete</b> - Delete the User Registry by name or ID.</li> <li>• <b>apic user-registries:execute</b> - Execute a User Registry operation.</li> <li>• <b>apic user-registries:get</b> - Get the User Registry by name or ID.</li> <li>• <b>apic user-registries:list</b> - List the User Registries.</li> <li>• <b>apic user-registries:search</b> - search for users in a user registry.</li> <li>• <b>apic user-registries:test-connection</b> - Test a User Registry connection.</li> <li>• <b>apic user-registries:update</b> - Update the User Registry by name or ID.</li> </ul>
apic user-registry-settings		<ul style="list-style-type: none"> <li>• <b>apic user-registry-settings:delete</b> - Delete the User Registry Setting.</li> <li>• <b>apic user-registry-settings:get</b> - Get the User Registry Setting.</li> <li>• <b>apic user-registry-settings:update</b> - Update the User Registry Setting.</li> </ul>
apic users		<ul style="list-style-type: none"> <li>• <b>apic users</b> - User collection operations.</li> <li>• <b>apic users:clear</b> - Clear the Users.</li> <li>• <b>apic users:create</b> - Create a User.</li> <li>• <b>apic users:delete</b> - Delete the User by name or ID.</li> <li>• <b>apic users:get</b> - Get the User by name or ID.</li> <li>• <b>apic users:list</b> - List the Users.</li> <li>• <b>apic users:request-password-reset</b> - send a password reset link.</li> <li>• <b>apic users:search-provider</b> - Search for provider users from an organization.</li> <li>• <b>apic users:update</b> - Update the User by name or ID.</li> </ul>



## Creating APIs and applications

You can develop API proxies and API implementations by using the developer toolkit. In the documentation, *API* refers to the API proxy and *application* refers to the API implementation.

LoopBack® is a high-performance Node.js interaction-tier framework for APIs and micro-services. The [LoopBack framework](#) can be downloaded and installed using the **CLI + LoopBack + Designer** option for downloading the toolkit. See [Installing the toolkit](#). To create a new LoopBack project, use the command `apic lb4`.

Note: IBM® API Connect is packaged with, and supports, LoopBack version 4. For more information on how to migrate your LoopBack 3 applications to LoopBack 4, see [LoopBack 4 Migration Guide](#).

You can use the developer toolkit or LoopBack to create language-independent APIs by using [OpenAPI](#) to proxy to an existing backend implementation or to augment applications that are developed in other languages or frameworks such as [Express®](#), Java™, [Swift](#), Go, and others.

## Creating development artifact definitions

Use the `apic create` command to create development artifacts, by using the following commands:

Command	Description
<code>apic create:api</code>	Create an OpenAPI definition.
<code>apic create:api --wsdl filename</code>	Create a SOAP API definition from a WSDL definition file, or a .zip file that contains the WSDL definition files for a service. The name and version of the generated API are obtained from the WSDL file. If you upload a .zip file, you can include in the .zip file an options file to specify additional directives. For details, see <a href="#">Using an options file when importing a WSDL service</a> .
<code>apic create:product</code>	Create an API Product definition.

Note: You can create an API or Product from an OpenAPI template file by using the `--template template-name` option.

You can also create Product and API definitions non-interactively by providing the `--title` option. This option sets several values that you can also customize with additional options; for example:

```
apic create:api --title Routes
apic create:product --title "Climb On"
```

You can also create the API and Product definitions at the same time:

```
apic create:api --title Routes --product "Climb On"
apic create:api --wsdl globalweather.wsdl --product "Weather Forecasting"
```

Alternatively, you can create APIs and then reference them when you create a new Product; for example:

```
apic create:api --title Routes
apic create:api --title Ascents
apic create:product --title "Climb On" --apis "routes.yaml ascents.yaml"
```

## Validating development artifact definitions

After you edit development artifacts or before you publish artifacts, best practice is to validate them; for example:

```
apic validate routes.yaml # Validate an API
apic validate climb-on.yaml # Validate the Product and its APIs
apic validate climb-on.yaml --product-only # Validate the Product only (do not validate the referenced APIs)
```

Note: If the OpenAPI file that defines your API uses a `$ref` field to reference a fragment of OpenAPI code that is defined in a separate file, the `$ref` field is replaced with the contents of the target file before the draft API is created with the `apic drafts:validate` command. For more information, see [Using \\$ref to reuse code fragments in your OpenAPI files](#).

## Developing LoopBack applications

After you've created a LoopBack application with the `apic lb4 app` command, you can add additional functionality to the application with the commands listed in the following table. Run these commands from the project root directory of the application. APIs created with LoopBack can be imported into API Manager.

Command	Description	Subcommands
---------	-------------	-------------

Command	Description	Subcommands
<code>apic lb4</code>	<p>Create LoopBack project and project artifacts. With no subcommand, creates a new LoopBack project.</p> <p>Note: To use the <code>apic lb4</code> command, you need to add a double dash, <code>--</code>, after the <code>lb4</code> subcommand to tell <code>apic</code> that the options behind the <code>--</code> are for the subcommand belonging to <code>lb4</code>. The second <code>--</code> is to signify the end of the options. For example,</p> <pre>\$ apic lb4 controller -- --help</pre>	<ul style="list-style-type: none"> <li>• <code>app</code> - Create a new LoopBack project (default)</li> <li>• <code>controller</code> - Add a controller to a LoopBack 4 application</li> <li>• <code>datasource</code> - Add a datasource to a LoopBack 4 application</li> <li>• <code>discover</code> - Discover models from relational databases</li> <li>• <code>example</code> - Download one of LoopBack 4 example projects</li> <li>• <code>extension</code> - Create a LoopBack 4 extension</li> <li>• <code>import-lb3-models</code> - Import one or more LoopBack 3 models to a LoopBack 4 application</li> <li>• <code>interceptor</code> - Generate interceptors</li> <li>• <code>model</code> - Add a new model to a LoopBack 4 application</li> <li>• <code>observer</code> - Generate life cycle observers for application start and stop</li> <li>• <code>openapi</code> - Generate controllers and models from OpenAPI specs</li> <li>• <code>relation</code> - Add a relation between two models in a LoopBack 4 application</li> <li>• <code>repository</code> - Add a new repository for a selected model to a LoopBack 4 application</li> <li>• <code>rest-crud</code> - Generate rest configs for model endpoints</li> <li>• <code>service</code> - Add a new remote or local service to a LoopBack 4 application</li> <li>• <code>update</code> - Check or update project dependencies of LoopBack modules</li> </ul>

## Publishing APIs and applications

To publish APIs and applications by using the developer toolkit of IBM® API Connect, you set configuration variables to define where you want to publish, log in to the target cloud platform, and then use the appropriate publishing commands.

See the following sections for more information:

- [Setting configuration variables](#)
- [Logging in to API Connect](#)
- [Publishing APIs](#)

## Setting configuration variables

The `apic config` command provides global and project-based configuration variables that specify the target Catalog for publishing APIs and applications. The values of these variables are stored in `~/ .apiconnect/config` (for global variables) and `project-dir/ .apiconnect` (for project variables). For a full list of configuration variables, see [Overview of the command-line tool](#).

Set the `catalog` configuration variable to the URI of an API Connect Catalog to define a default Catalog target for all commands managing Catalogs. The `catalog` URI has the form:

```
https://mgmt_endpoint_url/api/catalogs/org_name/catalog_name
```

where `mgmt_endpoint_url` is the platform API endpoint URL, `org_name` is the provider organization name, and `catalog_name` is the Catalog name. The `mgmt_endpoint_url` portion sets the default value of the `--server` option, the `org_name` portion sets the default value of the `--org` option, and the `catalog_name` portion sets the default value of the `--catalog` option; you can override any of these values by including the corresponding option in a command.

Set the `space` configuration variable to the URI of an API Connect Space, to define a default Space target for all commands that manage Spaces. The `space` URI has the form:

```
https://mgmt_endpoint_url/api/spaces/org_name/catalog_name/space_name
```

where `mgmt_endpoint_url` is the platform API endpoint URL, `org_name` is the provider organization name, `catalog_name` is the Catalog name, and `space_name` is the name of the Space. The `mgmt_endpoint_url` portion sets the default value of the `--server` option, the `org_name` portion sets the default value of the `--org` option, the `catalog_name` portion sets the default value of the `--catalog` option, and the `space_name` sets the default value of the `--space` option; you can override any of these values by including the corresponding option in a command.

Although setting these configuration variables is not required, doing so simplifies commands that interact with API Connect clouds by providing default values for frequently used command-line options.

Here is an example of publishing with and without the `catalog` configuration variable set.

Without the configuration variable set:

```
apic products publish climb-on.yaml --server mgmthost.com --org climbon --catalog sandbox
```

With the configuration variable set:

```
apic config:set catalog=https://platform-api.myserver.com/api/catalogs/climbon/sandbox
catalog: https://platform-api.myserver.com/api/catalogs/climbon/sandbox
apic products publish climb-on.yaml
```

You can override default values provided by the `catalog` configuration variable by providing one of the standard options with a different value. For example, use the `--catalog` option with the `apic products publish` command to specify the `qa` Catalog:

```
apic products publish climb-on.yaml --catalog qa
```

Don't forget about global configuration variables. If you use the same Catalog as the default target for multiple projects, set the value globally:

```
apic config:set --global catalog=https://platform-api.myserver.com/api/catalogs/climbon/sandbox
```

Note:

If you have an environment variable of the same name as a CLI configuration property then, by default, its value will override the value of the corresponding CLI configuration property for any CLI command at that scope.

For example, if you have defined an environment variable called `SPACE` then, by default, that value will be assumed for the value of the `--space` parameter in the following command, regardless of any `space` configuration property setting:

```
apic products:publish my_product.yaml --scope space
```

To prevent environment variables overriding CLI configuration properties, define an environment variable called `APIC_LOAD_FROM_ENV`, set to the value `false`.

## Logging in to API Connect

---

Use the `apic login` and `apic logout` commands to manage your authentication. For details, see [Logging in to a management server](#).

## Publishing APIs

---

Publishing APIs to API Catalogs in API Connect clouds enables you to socialize the APIs by using the developer portal and secure them by using the Gateway.

An *API Product* (or simply *Product*) is used to compose APIs for publishing. API Product managers can use it to bundle one or more APIs together, control the visibility of the Product in the developer portal (for example, only allow partners `x`, `y`, and `z` to view and subscribe to the Product), and define Plans to provide consumption options. The Products that reference the APIs and define the consumption Plans are also the primary unit of lifecycle management for APIs.

Use the `apic products publish` command (equivalent to `apic products:publish`) to publish API Products to an API Connect cloud. The following example demonstrates how to create APIs composed by a Product, and publishing the Product and its APIs to a Catalog:

```
apic create:api --title Routes
apic create:api --title Ascents
apic create:product --title "Climb On" --apis "routes.yaml ascents.yaml"
apic config:set catalog=https://platform-api.myserver.com/api/catalogs/climbon/sandbox
apic login --username some-user --password some-password --server platform-api.myserver.com --realm provider/default-idp-2
apic products publish climb-on.yaml
```

For full details on how to log in to your management server from the CLI, see [Logging in to a management server](#).

Add the `--stage` option to `apic publish` to stage the Product into a Catalog instead of publishing it. Products in a Catalog can be in the following states: staged, published, deprecated, or retired. For example:

```
apic products publish --stage climb-on.yaml
```

You can use a *Space* to partition a Catalog so multiple teams can manage Products and APIs independently in a single Catalog. A Space is conceptually like a subcatalog, except that Products and APIs in all Spaces in a given Catalog are published to the same developer portal. For more information about Spaces, see [Using syndication in IBM API Connect](#).

If default configuration values have been set for the Space, Catalog, organization and management server, the following publish command could be used:

```
apic products publish --scope space product.yaml
```

where *product* is the name of the product that you want to publish.

Note: If the OpenAPI file that defines your API uses a `$ref` field to reference a fragment of OpenAPI code that is defined in a separate file, the `$ref` field is replaced with the contents of the target file before the Product that contains the API is staged or published with the `apic publish` command. For more information, see [Using \\$ref to reuse code fragments in your OpenAPI files](#).

## Managing API Products

---

Use the `apic products` and `apic apis` commands to manage Products and APIs that have been published to IBM® API Connect Catalogs. Use the `--scope space` option to manage Products and APIs that have been published to Spaces within Catalogs.

## Product management command summary

---

The following table summarizes the commands available for managing API Products. Usage examples follow.

Example command	Description
<code>apic config:set catalog=https://platform-api.myserver.com/api/catalogs/myorg/sandbox</code>	Set the default Catalog.
<code>apic login --username some-user --password some-password --server platform-api.myserver.com --realm provider/my-identity-provider</code>	Login into the management server. For full details on how to log in to your management server from the CLI, see <a href="#">Logging in to the management server</a> .
<code>apic create:api --title Routes --product "ClimbOn"</code>	Create the Product and API.
<code>apic products:publish climbon.yaml</code>	Publish the Product to the default catalog. Add <code>--stage</code> argument to stage the Product.
<code>apic products:list-all --scope catalog</code>	List the Products in the default catalog.
<code>apic products:get climbon:1.0.0 --output -</code>	Display the Product's properties. Omit the output argument to download to a file.
<code>apic apis:list-all --scope catalog</code>	List the APIs in the Catalog.
<code>apic apis:get routes:1.0.0 --output -</code>	Get the API properties. Omit the output argument to download to a file.
<code>apic products:replace climbon:1.0.0 mapfile.txt</code>	Replace the Product indicated on the command line with the staged or published Product designated in the mapfile. This retires the replaced Product.
<code>apic products:supersede climbon:1.0.0 mapfile.txt</code>	Supersede the Product indicated on the command line with the staged or published Product designated in the mapfile. This deprecates the superseded Product.
<code>apic products:delete climbon:1.0.0</code>	Delete the Product from the Catalog. The Product must be either retired or deprecated.

## Using the `products` and `apis` commands through a full lifecycle

This example shows a complex lifecycle where a new version of a Product and API replaces the original version at run time.

Note: If the OpenAPI file that defines your API uses a `$ref` field to reference a fragment of OpenAPI code that is defined in a separate file, the `$ref` field is replaced with the contents of the target file before the Product that contains the API is staged or published with the `apic publish` command. For more information, see [Using \\$ref to reuse code fragments in your OpenAPI files](#).

Set the default Catalog and login to the mgmthost.com API Connect cloud

```
apic config:set catalog=https://platform-api.myserver.com/api/catalogs/climbon/sandbox
apic login --username some-user --password some-password --server platform-api.myserver.com --realm my-identity-provider
```

For full details on how to log in to your management server from the CLI, see [Logging in to the management server](#).

Create and publish an initial version

```
apic create:api --title Routes --version 1.0.0 --filename routes100.yaml
apic create:product --title "Climb On" --version 1.0.0 --apis routes100.yaml --filename climbon100.yaml
apic products:publish climbon100.yaml
```

Create a new version to fix a bug in the API, stage it to the Catalog

```
apic create:api --title Routes --version 1.0.1 --filename routes101.yaml
apic create:product --title "Climb On" --version 1.0.1 --apis routes101.yaml --filename climbon101.yaml
apic products:publish --stage climbon101.yaml
```

Inspect the Catalog

```
apic products:list-all --scope catalog
```

This produces a list of all Products in the catalog. Copy the ID of the `climbon:1.0.1` Product. The ID resembles the following example.

```
https://server/api/catalogs/3eca6046-3c27-4ad/ab8772bf-0f65-45/products/8f854623-bda1-4f9
```

Create a mapping file

For example, if you are replacing a Product, the mapping file specifies the Product that you want to replace, and the mapping of the Plans from the source Product to the target Product. For example, if you are replacing `climbon:1.0.0` with `climbon:1.0.1` then `product_url` property specifies the URL of `climbon:1.0.0`. This file takes the following form:

```
product_url: https://server/api/catalogs/{id}/products/{id}
plans:
- source: {source_plan_name_1}
 target: {target_plan_name_1}
- source: {source_plan_name_2}
 target: {target_plan_name_2}
 .
 .
 .
```

Note:

- The source and target Plan names can be the same or different.
- Every Plan in the Product you are replacing must be mapped to a Plan in the replacement Product.

"Hot-replace" version 1.0.0 with 1.0.1

```
apic products:replace climbon:1.0.1 PRODUCT_PLAN_MAPPING_FILE
```

After it is replaced in this way, a Product is retired. It is no longer active.

Note: The Product specified on the command line is the replacement Product. For example, if you are replacing `climbon:1.0.0` with `climbon:1.0.1` then the Product specified on the command line is `climbon:1.0.1`.

Supersede version 1.0.0 with 1.0.1

Rather than hot-replacing an existing Product with a new one (usually an updated version), you can supersede the existing Product with a new one. When you replace a Product, all external application subscriptions to that Product are automatically moved to the new Product. When you supersede a Product, the subscriptions are not automatically moved to the new Product, some action must be taken to subscribe external applications to the superseding Product. The command to supersede a Product uses the same mapping file as the command to replace a Product. The name of the superseding Product is given on the command line.

```
apic products:supersede climbon:1.0.1 PRODUCT_PLAN_MAPPING_FILE
```

Once superseded, a Product is marked deprecated, meaning that no further subscriptions to the Product are allowed, although it is still active and existing subscriptions still work.

Note: The Product specified on the command line is the superseding Product. For example, if you are replacing `climbon:1.0.0` with `climbon:1.0.1` then the Product specified on the command line is `climbon:1.0.1`. The mapping file specifies the URL of the Product you are superseding.

Set a migration target

It is possible to set a migration target for the existing Product. This helps migration.

Use a command like the following to set the migration target.

```
apic products:set-migration-target climbon:1.0.0 PRODUCT_PLAN_MAPPING_FILE
```

Note: The Product specified on the command line is the Product from which you want to migrate subscriptions. The mapping file specifies the target Product for the subscriptions.

Once a migration target is set, external application developers will be able to migrate their application subscriptions through the Developer Portal with ease. It is also possible to execute a subscription migration using the following command.

```
apic products:execute-migration-target climbon:1.0.0
```

Delete a Product

When the replaced or superseded Product is no longer needed, it can be deleted.

```
apic products:delete climbon:1.0.0
```

## Additional Product and API operations

In addition to the lifecycle management capabilities, you can download Products and APIs in a catalog or space using the `clone` sub-command:

Command	Description
<code>apic products:clone</code>	Download all Products and their APIs from the catalog or space.
<code>apic apis:clone</code>	Download all APIs from the catalog or space.

It can also be useful to clear all Products and their APIs from a Catalog, particularly for a development Catalog (you must provide the name of the Catalog as the value of the `--confirm` parameter):

```
apic products:clear --confirm catalog_name
```

where `catalog_name` is the name of the Catalog.

You can use a *Space* to partition a Catalog so multiple teams can manage Products and APIs independently in a single Catalog. A Space is conceptually like a subcatalog, except that Products and APIs in all Spaces in a given Catalog are published to the same developer portal. For more information about Spaces, see [Using syndication in IBM API Connect](#).

To manage the Products and APIs that have been published to a Space, include the `--scope space` option with the `apic products` and `apic apis` commands. For example, to list the Products that are contained in a Space called `flights`, use the following command:

```
apic products --scope space --space flights --catalog production --org climbonorg --server platform-api.myserver.com
```

## Changing the lifecycle state of a Product in a Catalog or Space

To directly change the lifecycle state of a Product that has previously been staged or published to a Catalog or Space, complete the following steps:

- Enter the following command (the terminating hyphen character means that the command takes input from the command line):

```
apic products:update product_name:version --server mgmt_endpoint_url --org organization --scope scope --catalog catalog [-space space] -
```

where:

- `product_name` is the name of the Product whose lifecycle state you want to change.
- `version` is the version of that Product.
- `mgmt_endpoint_url` is the platform API endpoint URL.
- `organization` is the name of the provider organization in which the Product was previously staged or published.
- `scope` has one of the following values:
  - `catalog` if the Catalog does not have Spaces enabled.
  - `space` if the Catalog has Spaces enabled. If you specify `space` for the `--scope` parameter you must also supply the `--space` parameter.
- `catalog` is the name of the Catalog in which the Product was previously staged or published.
- (optional) `space` is the name of the Space. The `--space` parameter is required if the Catalog has Spaces enabled, in which case you must also include `--scope space` in the command.

The command returns:

```
Reading PRODUCT_FILE arg from stdin
```

- Enter the following data, followed by a new line:

```
state: new_state
```

where `new_state` is the state that you want to change the Product to, and must have one of the following values.

- staged
- published
- deprecated
- retired

- archived
- Press **CTRL D** to terminate the input. If the command is successful, the lifecycle state change is confirmed.

For example:

```
apic products:update finance:1.0.0 --server https://myserver.com --org development --scope catalog --catalog sandbox -
Reading PRODUCT_FILE arg from stdin
state: published
finance:1.0.0 [state: published] https://myserver.com/api/catalogs/dce12994-a6a1-487b-83b6-c73bd8498799/006827d5-
9e82-427a-abe6-be5b126210f7/products/0f0af980-f505-4f36-b09c-d7b1c9c1a2f2
```

Note:

The command will not succeed if the lifecycle state change you are attempting is not permitted.

For example, the following lifecycle state changes are permitted:

- staged to published
- deprecated to retired

The following lifecycle state changes are not permitted:

- published to staged
- retired to published

For full details, see [The Product lifecycle](#).

To list all Products in a Catalog or Space, together with their lifecycle states, use the following command:

```
apic products:list-all --server mgmt_endpoint_url --org organization --scope scope --catalog catalog [--space space]
```

For example:

```
apic products:list-all --server https://myserver.com --org development --scope catalog --catalog sandbox
graphql-services:1.0.0 [state: staged] https://myserver.com/api/catalogs/dce12994-a6a1-487b-83b6-c73bd8498799/006827d5-
9e82-427a-abe6-be5b126210f7/products/7652d568-b396-4bfa-bf71-2f18cea63737
finance:1.0.0 [state: published] https://myserver.com/api/catalogs/dce12994-a6a1-487b-83b6-c73bd8498799/006827d5-
9e82-427a-abe6-be5b126210f7/products/0f0af980-f505-4f36-b09c-d7b1c9c1a2f2
```

To find out the lifecycle state of a specific Product, use the following command:

```
apic products:get product_name:version --server mgmt_endpoint_url --org organization --scope scope --catalog catalog [--space
space] --fields state --output -
```

For example:

```
apic products:get finance:1.0.0 --server https://myserver.com --org development --scope catalog --catalog sandbox --fields
state --output -
state: published
```

## Example scripts

A set of example toolkit scripts that demonstrate how to create and manage organizations, users, apps, Products and APIs are available at <https://github.com/ibm-apiconnect/example-toolkit-scripts>.

## Working with Drafts

Co-locate your APIs and applications in your local source code control systems to support typical development activities such as commits, branching, merges, continuous integration, and so on. The developer toolkit provides the bridge from the developer's environment to the IBM® API Connect runtime services.

API Connect provides an online development capability called *Drafts* where you can define API and Product definitions. The `apic drafts` commands enable synchronization of Product and API artifacts between local source code control systems and drafts.

You can use `drafts` to clear and list your drafts. For example:

```
apic config:set catalog=https://platform-api.myserver.com/api/catalogs/climbon/sandbox # set the default Catalog
apic login --username some-user --password some-password --server platform-api.myserver.com --realm provider/myldap # login
into the management server
```

```
apic create:api --title routes --product ClimbOn
apic drafts # list what is in drafts
apic drafts:clear --confirm drafts # clear drafts collection
apic drafts:list # list the available drafts
```

For full details on how to log in to your management server from the CLI, see [Logging in to the management server](#).

## Reading input from the command line

If a developer toolkit command takes a file as an input parameter, you can direct the command to read the input directly from the command line rather than supplying a separate file; this can be useful when writing scripts to automate command line operations, for example.

## Reading input from a file

Many developer toolkit commands that create or update new objects in API Connect take a file as an input parameter. For example:

- Register a new gateway service:

```
apic gateway-services:create [flags] GATEWAY_SERVICE_FILE
```

- Update a provider organization:

```
apic orgs:update [flags] ORG_FILE
```

- Create a new Catalog:

```
apic catalogs:create [flags] CATALOG_FILE
```

- Create a new consumer organization:

```
apic consumer-orgs:create [flags] CONSUMER_ORG_FILE
```

- Register a new developer application:

```
apic apps:create [flags] APP_FILE
```

- Create a new email server:

```
apic mail-servers:create [flags] MAIL_SERVER_FILE
```

- Update the list of applications that subscribe to a Product:

```
apic subscriptions:update [flags] SUBSCRIPTION_FILE
```

You can format the input file as a YAML file (the default) or as a JSON file. To use a JSON file as input, include the `--format json` parameter.

## Reading input from the command line

As an alternative to supplying an explicit file name to the command, you can supply a hyphen (-) in place of the file name parameter; the command will then read the file content directly from the command line as shown in the following examples.

- Register a new DataPower® Gateway (v5 compatible) gateway service:

```
apic gateway-services:create --server platform-api.myserver.com --org admin --availability-zone availability-zone-default -
Reading GATEWAY_SERVICE_FILE arg from stdin
name: dpgw-service
title: DataPower gateway service, compatible with v5
gateway_service_type: datapower-gateway
endpoint: 'https://mygwhost.com:3000'
api_endpoint_base: 'https://mygwhost.com:9443'
sni:
 - host: '*'
 tls_server_profile_url: https://platform-api.myserver.com/api/orgs/75203636-f038-4287-a732-24af4bf7059d/tls-server-
profiles/3c0a0e93-6aa4-4288-b09c-eccf4901b104
visibility:
 type: public
```

Note: You can obtain the `tls_server_profile_url` property value for a TLS server profile by using the following command, which lists the URLs for all TLS server profiles:

```
apic tls-server-profiles --org organization_name --server mgmt_endpoint_url
```

- Create a new provider organization:

```
apic orgs:create --server platform-api.myserver.com -
Reading ORG_FILE arg from stdin
name: development
title: Development organization
owner_url: https://platform-api.myserver.com/api/user-registries/1bbbd414-22f1-47cf-8eac-2050530d29a7/c082d755-866e-4394-
959f-fbb264c6c3a1/users/2db491c7-2d0f-4f60-a8ae-f38d07481f21
```

Note: You can obtain the `owner_url` property value by using the following command, which returns the `url` property for a specific user:

```
apic users:get username --user-registry user_registry_name --server mgmt_endpoint_url --org organization_name --fields url
--output -
```

- Create a new Catalog:

```
apic catalogs:create --server platform-api.myserver.com --org myorg -
Reading CATALOG_FILE arg from stdin
name: production
title: Production Catalog
summary: Catalog containing APIs in production use
```

- Create a new consumer organization:

```
apic consumer-orgs:create --server platform-api.myserver.com --catalog sandbox --org myorg -
Reading CONSUMER_ORG_FILE arg from stdin
name: finance-apps
title: Developers of finance applications
owner_url: https://platform-api.myserver.com/api/user-registries/a1fb8159-fda1-410f-93ab-a0ea5fd04535/70ce7ec3-e83f-4c87-
9f3c-4662ad108bbc/users/fddd5df5-c178-4a34-ab92-0a48344d5c9b
```

Note: You can obtain the `owner_url` property value by using the following command, which returns the `url` property for a specific user:

```
apic users:get username --user-registry user_registry_name --server mgmt_endpoint_url --org organization_name --fields url
--output -
```

- Register a new developer application:

```
apic apps:create --server consumer-api.myserver.com --org my-consumer-org --mode consumer -
Reading APP_FILE arg from stdin
name: finance
title: Finance application
summary: Mobile app for personal finance management
```

- Create a new email server:

```
apic mail-servers:create --org admin --server platform-api.myserver.com -
Reading MAIL_SERVER_FILE arg from stdin
title: My email server
name: my-email-server
host: smtp.myemail.com
port: 20
credentials:
 username: me@myemail.com
 password: password
```

- Create a new subscription to subscribe an application to a Product:

```
apic subscriptions:create --app myapp --catalog sandbox --org myorg --server platform-api.myserver.com --consumer-org my-
consumer-org -
Reading SUBSCRIPTION_FILE arg from stdin
product_url: https://platform-api.myserver.com/api/catalogs/960733c1-d7f7-4b90-9bc7-69dbf4ce31/ca34537d-adf4-4320-8495-
a7feaa62d679/products/638015c6-a8b0-452e-841f-2ac441ab6962
plan: default-plan
```

Note: You can obtain the `product_url` property value for a Product by using the following command, which lists the URLs for all versions of a Product given the Product name:

```
apic products:list product_name --scope catalog --catalog catalog_name --org provider_org_name --server mgmt_endpoint_url
```

---

## Related concepts

- [Overview of the command-line tool](#)

---

## Related reference

- [Logging in to a management server](#)

---

## Related information

- [Command-line tool reference](#)

---

# Scripting with the toolkit commands

The IBM® API Connect developer toolkit provides commands for cloud administration and API development and management.

---

## Scripting commands

It's often helpful to automate a series of `apic` commands in a shell script. Since the `apic` tool first requires you to interactively accept the license, you must first use the following command:

```
apic --accept-license
```

Once you do that, your scripts can run non-interactively.

Toolkit command scripts allow you to automate many of the tasks required to set up and maintain your API products. A set of example scripts are available at <https://github.com/ibm-apiconnect/example-toolkit-scripts>.

These scripts demonstrate how to perform the following tasks.

- Create a Provider Organization
- Configure a catalog to use one or more gateways
- Create draft APIs and Products
- Publish Products
- Replace an existing published product with a new version
- Delete a product
- Create a Consumer Organization
- Create a new consumer app
- Subscribe the new app to an existing published product
- Stage a new product version
- Supersede the existing published product with the new one
- Migrate existing subscriptions from the old product to the new product

---

## Related information

- [Command-line tool reference for the developer toolkit](#)



## Working with OpenAPI extensions

Use the developer toolkit CLI **extensions** commands to manage OpenAPI extensions in your Catalogs or Spaces.

You can extend the OpenAPI specification by adding either a JSON or YAML extension schema to an API, depending on the version of IBM® API Connect you are using. An extension is imported into a Catalog, or as Space ion a Catalog, then added to the API schema.

The commands described here are for importing OpenAPI extensions into a Catalog or Space, and viewing and managing them. After you have imported an OpenAPI extension, you can reference it in your API definitions; for details, see [Referring to an extension in an API definition](#).

Note: If Spaces are enabled in a Catalog, an OpenAPI extension that you import into one Space is imported into **all** Spaces; you cannot import an OpenAPI extension into an individual Space in the Catalog. Any subsequent updates are also applied to all Spaces.

Command name	Action	Syntax
<b>extensions clear</b>	<p>The <b>extensions:clear</b> command deletes all versions of an extension given the extension name.</p> <p>The parameters are as follows:</p> <p>--scope Specifies whether you want to clear the extensions in a Catalog or Space. The following options are available:</p> <p><b>catalog</b> The command is to be applied to Catalog</p> <p><b>space</b> The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to <b>space</b> and supply the <b>--space</b> parameter.</p> <p>--catalog or -c &lt;catalog_name&gt; Specifies a single Catalog with the Catalog name.</p> <p>--space Specifies the name of a Space in a Catalog. The <b>--space</b> parameter is required if the Catalog has Spaces enabled, in which case you must also include <b>--scope space</b> in the command.</p> <p>--configured-gateway-service &lt;service_name&gt; Specifies the name of the gateway service.</p> <p>--confirm &lt;catalog_name&gt; Confirms the clear of the extension.</p> <p>&lt;extension_name&gt; Specifies the name of the existing extension. Note: This is the extension name, not the file name.</p> <p>--org or -o &lt;organization_name&gt; Specifies a single organization with the organization name.</p> <p>--server or -s &lt;management_server_endpoint&gt; Specifies the server endpoint.</p>	<pre>apic extensions:clear --scope catalog space --catalog   -c catalog_name extension_name --server   -s management_server_endpoint --configured-gateway-service gw_service_name --confirm catalog_name --org   -o org_name [--space space]</pre> <p>Example:</p> <pre>apic extensions:clear --scope catalog --catalog catalog1 extension1 --server endpoint1 --configured-gateway-service gw_service1 --confirm catalog1 --org my_org</pre> <p>This example deletes all versions of the <b>extension1</b> extension that are in <b>catalog1</b> of the <b>my_org</b> organization, and are paired with <b>endpoint1</b>.</p>
<b>extensions clear-all</b>	<p>The <b>extensions:clear-all</b> command deletes all of the extensions in the specified Catalog or Space.</p> <p>The parameters are as follows:</p> <p>--scope Specifies whether you want to clear the extensions in a Catalog or Space. The following options are available:</p> <p><b>catalog</b> The command is to be applied to Catalog</p> <p><b>space</b> The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to <b>space</b> and supply the <b>--space</b> parameter.</p> <p>--catalog or -c &lt;catalog_name&gt; Specifies a single Catalog with the Catalog name.</p> <p>--space Specifies the name of a Space in a Catalog. The <b>--space</b> parameter is required if the Catalog has Spaces enabled, in which case you must also include <b>--scope space</b> in the command.</p> <p>--configured-gateway-service &lt;service_name&gt; Specifies the name of the gateway service.</p> <p>--confirm &lt;catalog_name&gt; Confirms the clear of the extension.</p> <p>--org or -o &lt;organization_name&gt; Specifies a single organization with the organization name.</p> <p>--server or -s &lt;management_server_endpoint&gt; Specifies the server endpoint.</p>	<pre>apic extensions:clear-all --scope catalog space --catalog   -c catalog_name --server   -s mgt_server_endpoint --configured-gateway-service gw_service_name --confirm catalog_name --org   -o org_name [--space space]</pre> <p>Example:</p> <pre>apic extensions:clear-all --scope catalog --catalog catalog1 --server endpoint1 --configured-gateway-service gw_service1 --confirm catalog1 --org my_org</pre> <p>This example deletes all the extensions that are in <b>catalog1</b> of the <b>my_org</b> organization, and are paired with <b>endpoint1</b>.</p>

Command name	Action	Syntax
<b>extensions clone</b>	<p>The <b>extensions:clone</b> command creates a local definition file for each extension.</p> <p>The parameters are as follows:</p> <p>--scope Specifies whether you want to clone the extensions in a Catalog or Space. The following options are available:</p> <p><b>catalog</b> The command is to be applied to Catalog</p> <p><b>org</b> Sets the scope to the organization.</p> <p><b>space</b> The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to <b>space</b> and supply the <b>--space</b> parameter.</p> <p>--catalog or -c &lt;catalog_name&gt; Specifies a single Catalog with the Catalog name.</p> <p>--space Specifies the name of a Space in a Catalog. The <b>--space</b> parameter is required if the Catalog has Spaces enabled, in which case you must also include <b>--scope space</b> in the command.</p> <p>--org or -o &lt;organization_name&gt; Specifies a single organization with the organization name.</p> <p>--server or -s &lt;management_server_endpoint&gt; Specifies the server endpoint.</p> <p>--configured-gateway-service &lt;service_name&gt; Specifies the name of the gateway service.</p>	<pre>apic extensions:clone --scope catalog space --catalog -c catalog_name --org -o org_name [--space space] --server -s mgt_server_endpoint --configured-gateway-service gw_service_name</pre> <p>Example:</p> <pre>apic extensions:clone --scope catalog --catalog catalog1 --org my_org --server endpoint1 --configured-gateway-service gw_service_1</pre> <p>This example clones the extensions that are in <b>catalog1</b> of the <b>my_org</b> organization, and are paired with <b>endpoint1</b>.</p>
<b>extensions create</b>	<p>The <b>extensions:create</b> command creates an extension in a Catalog or Space, given an extension definition file.</p> <p>The parameters are as follows:</p> <p>--scope Specifies whether you want to clear the extensions in a Catalog or Space. The following options are available:</p> <p><b>catalog</b> The command is to be applied to Catalog</p> <p><b>space</b> The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to <b>space</b> and supply the <b>--space</b> parameter.</p> <p>&lt;extension_file&gt; Specifies the name of the extension file that contains the new information for your extension. Note: This is the OpenAPI file name, not the name of the extension.</p> <p>--server or -s &lt;management_server_endpoint&gt; Specifies the server endpoint.</p> <p>--catalog or -c &lt;catalog_name&gt; Specifies a single Catalog with the Catalog name.</p> <p>--space Specifies the name of a Space in a Catalog. The <b>--space</b> parameter is required if the Catalog has Spaces enabled, in which case you must also include <b>--scope space</b> in the command.</p> <p>--org or -o &lt;organization_name&gt; Specifies a single organization with the organization name.</p> <p>--configured-gateway-service &lt;service_name&gt; Specifies the name of the gateway service.</p>	<pre>apic extensions:create --scope catalog space extension_file --server -s mgt_server_endpoint --catalog -c catalog_name [--space space] --org -o org_name --configured-gateway-service gw_service_name</pre> <p>Example:</p> <pre>apic extensions:create --scope catalog filename.yaml --server endpoint1 --catalog catalog1 --org my_org --configured-gateway-service gw_service_1</pre> <p>This example creates an extension in <b>catalog1</b> of the <b>my_org</b> organization, paired with <b>endpoint1</b>, using the <b>filename.yaml</b> OpenAPI extension definition file.</p>

Command name	Action	Syntax
<b>extensions delete</b>	<p>The <b>extensions:delete</b> command deletes a specific version of an extension.</p> <p>The parameters are as follows:</p> <p>--scope Specifies whether you want to clear the extensions in a Catalog or Space. The following options are available:</p> <p><b>catalog</b> The command is to be applied to Catalog</p> <p><b>space</b> The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to <b>space</b> and supply the <b>--space</b> parameter.</p> <p>--server or -s &lt;management_server_endpoint&gt; Specifies the server endpoint.</p> <p>--catalog or -c &lt;catalog_name&gt; Specifies a single Catalog with the Catalog name.</p> <p>--space Specifies the name of a Space in a Catalog. The <b>--space</b> parameter is required if the Catalog has Spaces enabled, in which case you must also include <b>--scope space</b> in the command.</p> <p>&lt;extension_name&gt;:&lt;version_number&gt; Specifies the name and version number of the extension. Note: <i>extension_name</i> is the extension name, not the file name.</p> <p>--org or -o &lt;organization_name&gt; Specifies a single organization with the organization name.</p> <p>--configured-gateway-service &lt;service_name&gt; Specifies the name of the gateway service.</p>	<pre>apic extensions:delete --scope catalog space   extension_name:version_number   --catalog -c catalog_name   --org -o org_name [--space space]   --server -s mgt_server_endpoint   --configured-gateway-service gw_service_name</pre> <p>Example:</p> <pre>apic extensions:delete --scope catalog myextension:1.0.0 --catalog catalog1 --org orgmain --server endpoint1 --configured-gateway-service gw_service_1</pre> <p>This example deletes <b>myextension</b> version 1.0.0 that is in <b>catalog1</b> of the <b>orgmain</b> organization, and is paired with <b>endpoint1</b>.</p>
<b>extensions get</b>	<p>The <b>extensions:get</b> command retrieves the definition file for a specific version of an extension.</p> <p>The parameters are as follows:</p> <p>--scope Specifies whether you want to clone the extensions in a Catalog or Space. The following options are available:</p> <p><b>catalog</b> The command is to be applied to Catalog</p> <p><b>org</b> Sets the scope to the organization.</p> <p><b>space</b> The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to <b>space</b> and supply the <b>--space</b> parameter.</p> <p>--server or -s &lt;management_server_endpoint&gt; Specifies the server endpoint.</p> <p>--catalog or -c &lt;catalog_name&gt; Specifies a single Catalog with the Catalog name.</p> <p>--space Specifies the name of a Space in a Catalog. The <b>--space</b> parameter is required if the Catalog has Spaces enabled, in which case you must also include <b>--scope space</b> in the command.</p> <p>&lt;extension_name&gt;:&lt;version_number&gt; Specifies the name and version number of the extension. Note: <i>extension_name</i> is the extension name, not the file name.</p> <p>--org or -o &lt;organization_name&gt; Specifies a single organization with the organization name.</p> <p>--configured-gateway-service &lt;service_name&gt; Specifies the name of the gateway service.</p>	<pre>apic extensions:get --scope catalog space   extension_name:version_number   --catalog -c catalog_name   --org -o org_name [--space space]   --server -s mgt_server_endpoint   --configured-gateway-service gw_service_name</pre> <p>Example:</p> <pre>apic extensions:get --scope catalog myextension:1.0.0 --catalog catalog1 --org orgmain --server endpoint1 --configured-gateway-service gw_service_1</pre> <p>This example gets <b>myextension</b> version 1.0.0 that is in <b>catalog1</b> of the <b>orgmain</b> organization, and is paired with <b>endpoint1</b>.</p>

Command name	Action	Syntax
<b>extensions list</b>	<p>The <b>extensions:list</b> command lists all versions of an extension given the extension name.</p> <p>The parameters are as follows:</p> <p>--scope Specifies whether you want to clone the extensions in a Catalog or Space. The following options are available:</p> <p><b>catalog</b> The command is to be applied to Catalog</p> <p><b>org</b> Sets the scope to the organization.</p> <p><b>space</b> The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to <b>space</b> and supply the <b>--space</b> parameter.</p> <p>--server or -s &lt;management_server_endpoint&gt; Specifies the server endpoint.</p> <p>--catalog or -c &lt;catalog_name&gt; Specifies a single Catalog with the Catalog name.</p> <p>--space Specifies the name of a Space in a Catalog. The <b>--space</b> parameter is required if the Catalog has Spaces enabled, in which case you must also include <b>--scope space</b> in the command.</p> <p>&lt;extension_name&gt; Specifies the name of the existing extension. Note: This is the extension name, not the file name.</p> <p>--org or -o &lt;organization_name&gt; Specifies a single organization with the organization name.</p> <p>--configured-gateway-service &lt;service_name&gt; Specifies the name of the gateway service.</p>	<pre>apic extensions:list --scope catalog space extension_name --catalog   -c catalog_name --org   -o org_name [--space space] --server   -s mgt_server_endpoint --configured-gateway-service gw_service_name</pre> <p>Example:</p> <pre>apic extensions:list --scope catalog my_extension --catalog catalog1 --org my_org --server endpoint1 --configured-gateway-service gw_service_1</pre> <p>This example lists all versions of the extension <b>my_extension</b> that are paired with <b>endpoint1</b>, are in <b>catalog1</b>, and in the <b>my_org</b> organization.</p>
<b>extensions list-all</b>	<p>The <b>extensions:list-all</b> command lists all of the extensions that are available in a Catalog or Space. This is the default command if you enter only <b>apic extensions</b>.</p> <p>The parameters are as follows:</p> <p>--scope Specifies whether you want to clone the extensions in a Catalog or Space. The following options are available:</p> <p><b>catalog</b> The command is to be applied to Catalog</p> <p><b>org</b> Sets the scope to the organization.</p> <p><b>space</b> The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to <b>space</b> and supply the <b>--space</b> parameter.</p> <p>--server or -s &lt;management_server_endpoint&gt; Specifies the server endpoint.</p> <p>--catalog or -c &lt;catalog_name&gt; Specifies a single Catalog with the Catalog name.</p> <p>--space Specifies the name of a Space in a Catalog. The <b>--space</b> parameter is required if the Catalog has Spaces enabled, in which case you must also include <b>--scope space</b> in the command.</p> <p>--org or -o &lt;organization_name&gt; Specifies a single organization with the organization name.</p> <p>--configured-gateway-service &lt;service_name&gt; Specifies the name of the gateway service.</p>	<pre>apic extensions:list-all --scope catalog space --catalog   -c catalog_name --org   -o org_name [--space space] --server   -s mgt_server_endpoint --configured-gateway-service gw_service_name</pre> <p>Example:</p> <pre>apic extensions:list-all --scope catalog --catalog catalog1 --org my_org --server endpoint1 --configured-gateway-service gw_service1</pre> <p>This example lists the extensions that are paired with <b>endpoint1</b>, are in <b>catalog1</b>, and in the <b>my_org</b> organization.</p>

Command name	Action	Syntax
<b>extensions update</b>	<p>The <b>extensions:update</b> command replaces the information of an existing version of an extension with the information in an external file.</p> <p>The parameters are as follows:</p> <p>--scope Specifies whether you want to clear the extensions in a Catalog or Space. The following options are available:</p> <p><b>catalog</b> The command is to be applied to Catalog</p> <p><b>space</b> The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to <b>space</b> and supply the <b>--space</b> parameter.</p> <p>--server or -s &lt;management_server_endpoint&gt; Specifies the server endpoint.</p> <p>--catalog or -c &lt;catalog_name&gt; Specifies a single Catalog with the Catalog name.</p> <p>--space Specifies the name of a Space in a Catalog. The <b>--space</b> parameter is required if the Catalog has Spaces enabled, in which case you must also include <b>--scope space</b> in the command.</p> <p>--org or -o &lt;organization_name&gt; Specifies a single organization with the organization name.</p> <p>--configured-gateway-service &lt;service_name&gt; Specifies the name of the gateway service.</p> <p>&lt;extension_name&gt;:&lt;version_number&gt; Specifies the name and version number of the extension. Note: <i>extension_name</i> is the extension name, not the file name.</p> <p>&lt;extension_file&gt; Specifies the name of the extension file that contains the new information for your extension. Note: This is the OpenAPI file name, not the name of the extension.</p>	<pre> <b>apic extensions:update</b> <b>--scope</b> catalog space <b>extension_name:version_number</b> <b>extension_file</b> <b>--catalog   -c</b> catalog_name <b>--org   -o</b> org_name <b>[--space space]</b> <b>--server   -s</b> mgt_server_endpoint <b>--configured-gateway-service</b> gw_service_name </pre> <p>Example:</p> <pre> <b>apic extensions:update</b> <b>--scope</b> catalog <b>extension1:1.0.0</b> extension2.yaml <b>--catalog</b> catalog1 <b>--org</b> my_org <b>--server</b> endpoint1 <b>--configured-gateway-service</b> gw_service1 </pre> <p>This example updates <b>extension1</b> version 1.0.0 that is paired with <b>endpoint1</b>, in <b>catalog1</b>, and in the <b>my_org</b> organization, from the file <b>extension2.yaml</b>.</p>

## Managing platform REST API keys

Create API keys associated with your user account, and then use them to authenticate when you call the platform REST APIs provided by API Connect.

### About this task

The [API Connect platform REST APIs](#) require authentication for use. To call an API interactively, you can log into the toolkit CLI with an option to generate the API key. To call an API programmatically, you can generate an API key and then exchange it for an API Connect access token. This requirement applies only to the APIs that are provided by API Connect. You do not need the API key for calling APIs that you imported or created. The API key is intended for use with the IBMid and might not work with other user registries.


### Procedure

- Create an API key:
  - Log in to API Connect.
  - On the address bar, add **/apikey** to the end of the URL (after the organization name).  
For example:  


```
https://manager.example.com/manager/myorg/apikeys
```
  - On the API Keys page, click Add.
  - In the "Create API Connect API Key" box, provide a title for the key (unique within your user account) and an optional description; then click Create.
- Manage API Keys:
  - Log in to API Connect.
  - On the address bar, add **/apikey** to the end of the URL.

## Searching for items in API Manager

Use the search feature in IBM® API Connect API Manager to easily locate items such as APIs, Catalogs, Spaces, applications, and subscriptions.

Search is available from any page in API Manager and can be accessed by typing in the search field in the page banner, or by clicking  Search in the navigation list.

### Using Search

The Search feature is available on all pages of the API Manager. To start searching, just type a string into the Search field and press Enter. Whenever you press Enter in the Search field, the Search page displays with results and additional options. If you press Enter in an empty Search field, or click  Search in the navigation list, then the Search page displays a list all items. If you type a string before pressing Enter, the page displays the results from the corresponding search.

Search results include all items that contain the search string in a text field. For example, suppose you searched for "Crafted". The results might look like the following image, with items that don't match the search string exactly, but do include the string in a text field such as a title or a name.

By default, the search returns all types of items that contain the string. Sort results on a particular column by clicking the column header.

When you find an item you want to work with, click the . . . actions menu next to the item's "Modified" date to see what you can do. The list of possible actions depends on the item's type.

You can limit the search results by selecting an item type from the list that displays before the results. Click More to view additional types. If you select a type from the More menu, it replaces a type in the list.

If you want to refine the search further, click Options. On the Options panel, click Catalogs to Search and select which catalogs to search (the default is all catalogs). You can select up to 5 catalogs for a limited search.

You can also choose whether only exact matches (the complete string) are included in results, or items containing similar strings are also included.

---

## Working with API definitions

An API definition specifies the complete configuration for an API. You can create and configure your APIs either by using either the API Designer UI application, or by using the browser based API Manager UI.

If you are working with APIs that conform to the OpenAPI Specification, note that API Connect uses the following terms:

- **OpenAPI Specification:**  
The OpenAPI Specification defines a standard, language-agnostic interface to RESTful APIs. API Connect supports version 2 and version 3 of the OpenAPI Specification.
- **OpenAPI document:**  
One or more files that describe an API conforming to the OpenAPI Specification.
- **OpenAPI definition:**  
The description of an API that conforms to the OpenAPI Specification. In API Connect, you can create the API definition by filling in settings and properties, or by importing an OpenAPI document that contains an API definition that you want to work with.

The following topics provide full details for creating and configuring your APIs.

- **[Creating an API definition](#)**  
You can create API definitions by using the API Designer or the command line interface in IBM® API Connect.
- **[Editing an OpenAPI 2.0 API definition](#)**  
IBM API Connect provides a form based editor that enables you to configure APIs that conform to the OpenAPI 2.0 specification.
- **[Editing an OpenAPI 3.0 API definition](#)**  
IBM API Connect provides a form based editor that enables you to configure APIs that conform to the OpenAPI 3.0 specification.
- **[LDAP authentication](#)**  
The Lightweight Directory Access Protocol (LDAP) is an internet protocol for accessing and maintaining distributed directory information services over a network. If you rely on LDAP to authenticate users for web applications, take a minute to review the contents of this topic before beginning.
- **[Authentication URL user registry](#)**  
You can use an Authentication URL user registry to specify a REST authentication service to manage user authentication, and to optionally provide additional metadata to be embedded in the token.
- **[Activating an API](#)**  
After you have created an API definition you can activate it to make it available for testing.
- **[Testing an API](#)**  
Select a test tool and verify that your API works as you intended.
- **[Staging an API](#)**  
The graphical wizard provides an option that adds the API to a Product and stages the Product in a Catalog. When a Product is in the staged state, it is not yet visible to, or subscribable by, any developers. The syndication feature in IBM API Connect means that if Spaces are enabled for a Catalog, Products can be staged only to a Space within that Catalog.
- **[Publishing an API](#)**  
The user interface provides an option that adds an API to a Product and publishes the Product in a Catalog. Publishing a draft Product makes the APIs in that Product visible on the Developer Portal for use by application developers. The syndication feature in IBM API Connect means that if Spaces are enabled for a Catalog, Products can be published only to a Space within that Catalog.
- **[Creating a new version of an API definition](#)**  
You can create multiple versions of an API definition and edit the versions independently by using IBM API Connect.
- **[Updating an API definition from a file](#)**  
You can update an API by uploading a .yaml or .json file that contains the OpenAPI definition for the revised API.
- **[Updating a SOAP API](#)**  
You can update the configuration of an existing SOAP API either by uploading a WSDL file, or by uploading a .zip archive that contains a primary WSDL file together with other WSDL or XSD files that it references.
- **[Renaming an API definition](#)**  
By renaming an API definition, you change the **name** property that, together with the version, is used to uniquely identify the API definition in API Connect. The **name** property is initially generated automatically when you first create the API definition, based on the title, but you can later change it if desired.
- **[Changing an API rate limit](#)**  
When you activate an API to make it available for testing, as described in [Activating an API](#), no rate limit is applied to restrict calls to the API. However, if desired, you can apply a rate limit to the API, or modify a previously applied rate limit. The rate limit defines the maximum number of calls allowed in a specified time period.

- [Adding an OpenAPI extension to an API](#)  
You can extend the OpenAPI specification by adding either a JSON or YAML extension schema to an API. An OpenAPI extension is imported into a Catalog, or to a Space within a Catalog, then added to the API schema.
- [Downloading an API definition](#)  
You can download the details of your API definitions so that you can store them for later recovery.
- [Deleting an API definition](#)  
You can delete an API definition that is no longer required.
- [Using an options file when importing a WSDL service](#)  
When you create an API definition, or add a target WSDL service to an API definition, by importing a .zip file, you can specify additional directives by including an options file in the .zip file.
- [Variable references in API Connect](#)  
In API Connect you can reference different variables in your API definition.
- [API properties](#)  
Properties are used by the gateway to control behavior of certain policies. Typically, you provide properties, but the policy can also provide properties settings.
- [API policies and logic constructs](#)  
Policies and logic constructs are pieces of configuration that control a specific aspect of processing in the Gateway server during the handling of an API invocation at run time.
- [Tags in API Connect](#)  
There are a number of forms of tagging in IBM API Connect.

---

## Creating an API definition

You can create API definitions by using the API Designer or the command line interface in IBM® API Connect.

### Before you begin

---

An API is a set of functions that provide some business or technical capability and can be called by applications by using a defined protocol. Applications are typically mobile or web applications, and they use the HTTP protocol. An API definition is composed of paths, and can be one of the following types:

#### REST API definition

A REST API is a defined set of interactions that uses the HTTP protocol, typically by using JSON or XML as the data format that is exchanged. For example, a data request might use an HTTP GET method, and a data record might use an HTTP POST method. The choice of data format depends on the type of application that is calling the API. JSON is commonly used for web pages or mobile applications that present a user interface (by using JavaScript or HTML), whereas XML is often used for machine-to-machine scenarios.

You can create REST APIs by using LoopBack® functions to create models and data sources. These are then used by your REST API definition and exposed to your users.

Alternatively, you can expose and secure your existing APIs by using a [Proxy](#) or [Invoke](#) policy.

In either case, you can configure your API definition either by using the API Designer, or by writing an OpenAPI definition file and publishing it using either API Designer or the command line interface.

#### SOAP API

You can create SOAP API definitions that are based on an existing Web Services Description Language (WSDL) file. You can use this facility to benefit from the capabilities that are provided by API Connect, which include analytics and mapping between variables. You can also expose the API by using the Developer Portal for any existing SOAP services in your organization, including any SOAP services that are part of a service-oriented architecture (SOA) or Enterprise Service Bus (ESB) infrastructure.

You can create SOAP API definitions through either the command line interface, or through API Designer.

### About this task

---

You can create definitions for REST APIs or SOAP APIs.

- In the API Designer, you can add a REST API definition either by composing the API definition, and its operations, from scratch, or by importing an OpenAPI definition file. You can also use the tooling to quickly create a proxy API that calls an existing endpoint.
- If you have an existing SOAP service that you want to expose more widely, you can add a SOAP API to IBM® API Connect. You can use the Developer Portal to publicize the SOAP service to the developers. If a developer wants to use the SOAP API, you can use API Connect to manage their sign-up and access to the service, and track the usage of that API.

### Procedure

---

1. Quickly create an API definition by using one of the following wizards:
    - [Creating a REST proxy API from a target service](#)
    - [Creating a REST proxy API from an existing OpenAPI service](#)
    - [Creating a new REST OpenAPI definition](#)
    - [Creating a REST proxy API from an existing WSDL service](#)
    - [Adding a REST API by importing an OpenAPI definition file](#)
    - [Creating a SOAP proxy API from an existing WSDL service](#)
  2. After you create your API definition, you can define it in more detail by following the instructions in [Editing an OpenAPI 2.0 API definition](#).
- [Creating a REST proxy API from a target service](#)  
If you have an existing REST service that you want to expose through an IBM API Connect API definition, you can create a proxy API and specify the target endpoint by using the API Designer.
  - [Creating a REST proxy API from an existing OpenAPI service](#)  
If you have an existing OpenAPI described target service that you want to expose through an IBM API Connect API definition, you can create a proxy API and

specify the target endpoint.

- [Creating a REST proxy API from an existing WSDL service](#)

If you have a SOAP service defined in a WSDL file, you can use the WSDL file to create an API Connect REST proxy API that calls that SOAP service.

- [Creating a SOAP proxy API from an existing WSDL service](#)

If you have a SOAP service defined in a WSDL file, you can use the WSDL file to create an API Connect SOAP proxy API that calls that SOAP service.

- [Creating a new REST OpenAPI definition](#)

You can create and edit draft REST API definitions by using the API Designer or the API Designer user interface in IBM API Connect.

- [Adding a REST API by importing an OpenAPI definition file](#)

You can use an OpenAPI definition file to import a REST API into IBM API Connect.

- [Adding a SOAP API by importing a zip file](#)

You can use an OpenAPI definition file to import a SOAP API into IBM API Connect.

- [Creating a GraphQL proxy API](#)

GraphQL is a query language for APIs that gives an application client greater control over what data it retrieves in an API request when compared with a REST API request. IBM API Connect enables you to create a GraphQL API proxy definition that proxies a backend GraphQL server, and to define rate limiting controls that reflect the amount of data that is returned from the server by a request to the GraphQL API.

- [Importing an API from the CP4I asset repository](#)

If you are using IBM API Connect with IBM Cloud Pak for Integration (CP4I), you can import an API definition from the CP4I asset repository.

## Related concepts

---

- [Using the developer toolkit command-line tool](#)

---

## Creating a REST proxy API from a target service

If you have an existing REST service that you want to expose through an IBM® API Connect API definition, you can create a proxy API and specify the target endpoint by using the API Designer.

### About this task

---

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.


To complete this task, you must be assigned a role that has the **Api-Drafts:Edit**, **Settings:View**, and **App:View** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see [Creating custom roles](#).

Create proxy REST APIs in minutes by using the API Designer tool to create an API from target service. This procedure creates a REST proxy that routes all traffic to a target REST API or service endpoint.

### Procedure

---

To compose a proxy API from a target service, complete the following steps.

1. In the navigation pane, click  Develop, select the APIs tab, then click Add API (from REST, GraphQL or SOAP). The Select API type screen is displayed.
2. Select OpenAPI 2.0 or OpenAPI 3.0 according to which version of the OpenAPI specification your API is to be based on.
3. Select From target service.
4. Click Next. Specify the API summary in the Info section. You can fine-tune the API after it is created.
  - The Title can include special characters but should be kept short so that it can be easily displayed in the user interface.
  - The Name is entered automatically. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see [apic draft-apis](#).
  - The Version corresponds to the value of the `info.version` property of the API's OpenAPI definition. The `version.release.modification` version numbering scheme is recommended; for example `1.0.0`.
  - The Base path is the URL segment of the API and does not include the host name or any additional segments for paths or operations. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
  - The optional Description helps to identify the API.
5. Specify the existing REST API endpoint that you want to call in the Target Service URL field.
6. Click Next. In the Secure section, configure the API security that you require.
  - Secure using Client ID - Select this option to require an Application to provide a Client ID (API Key). This causes the `X-IBM-Client-Id` parameter to be included in the request header for the API. If selected, you can then select whether to limit the API calls on a per key (per Client ID) basis:
    - Limit API calls on a per key basis - If selected, you must configure the rate limit that you require. Rate limits control the maximum number of calls allowed within a time period (hour, minute, month or day). For example, 100 calls per hour.For information about security options in IBM API Connect, see [Configuring API security](#).
  - CORS - Select this option to enable cross-origin resource sharing (CORS) support for your API. This allows your API to be accessed from another domain.

Note:

    - When CORS is enabled, the API Gateway runs the `cors` preflow policy to handle all CORS requests that are made to the API.
    - When CORS is enabled and a preflight request is received, only the following API actions are performed:
      - The `cors` preflow policy configures the appropriate response headers.
      - The response headers are set.
    - When a preflight request is received, the `request.attributes.isCORSPreflight` flag is set to `true`.
    - For all preflight requests, the `security` and `client-identification` preflow policies are always skipped, whether CORS is enabled or not enabled.
7. Click Next to create your API definition.

The Summary panel displays messages as the definition is created, and the selected security options and rate limits are enforced.
8. Select one of the following options:
  - To further configure your API, click Edit API. For details, see [Editing an API definition](#).



- If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see [Editing an API definition](#).

## Results

---

You created a proxy API from an existing target service.

## What to do next

---

APIs are made available to application developers by including them in a Product, and then publishing that Product to a Catalog. For more information, see [Working with Products](#) and [Working with Catalogs](#).

## Related tasks

---

- [Defining Paths for a REST API](#)

## Related reference

---

- [API and Product definition template examples](#)

# Creating a REST proxy API from an existing OpenAPI service

If you have an existing OpenAPI described target service that you want to expose through an IBM® API Connect API definition, you can create a proxy API and specify the target endpoint.

## About this task

---

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.



To complete this task, you must be assigned a role that has the **Api-Drafts:Edit**, **Settings:View**, and **App:View** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see [Creating custom roles](#).

Create proxy REST APIs in minutes by using the API Designer tool to create an API from existing OpenAPI service. This procedure creates a REST proxy that routes all traffic to a target REST API or service endpoint.

## Procedure

---

To compose a proxy API from an existing OpenAPI service, complete the following steps.

1. In the navigation pane, click  Develop, select the APIs tab, then click Add API (from REST, GraphQL or SOAP). The Select API type screen is displayed.
2. Select OpenAPI 2.0 or OpenAPI 3.0 according to which version of the OpenAPI specification your API is to be based on.
3. Select From existing OpenAPI service, and click Next.
4. To upload the service information from an OpenAPI file, you can either drag and drop your file, or browse and select the file that you want to use. Valid file types are YAML, YML, etc.
  - Click Browse and browse for the OpenAPI file.
  - Drag and drop the file into the active area.After you upload the file, the OpenAPI specification is evaluated and a message displays whether it is valid.
5. Click Next. Specify the API summary in the Info section. You can fine-tune the API after it is created.
  - The Title can include special characters but should be kept short so that it can be easily displayed in the user interface.
  - The Name is entered automatically. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see [apic draft-apis](#).
  - The Version corresponds to the value of the `info.version` property of the API's OpenAPI definition. The `version.release.modification` version numbering scheme is recommended; for example `1.0.0`.
  - The Base path is the URL segment of the API and does not include the host name or any additional segments for paths or operations. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
  - The optional Description helps to identify the API.
6. Click Next. In the Secure section, configure the API security that you require.
  - Secure using Client ID - Select this option to require an Application to provide a Client ID (API Key). This causes the `X-IBM-Client-Id` parameter to be included in the request header for the API. If selected, you can then select whether to limit the API calls on a per key (per Client ID) basis:
    - Limit API calls on a per key basis - If selected, you must configure the rate limit that you require. Rate limits control the maximum number of calls allowed within a time period (hour, minute, month or day). For example, 100 calls per hour.For information about security options in IBM API Connect, see [Configuring API security](#).
  - CORS - Select this option to enable cross-origin resource sharing (CORS) support for your API. This allows your API to be accessed from another domain.  
Note:
    - When CORS is enabled, the API Gateway runs the `cors` preflow policy to handle all CORS requests that are made to the API.
    - When CORS is enabled and a preflight request is received, only the following API actions are performed:
      - The `cors` preflow policy configures the appropriate response headers.
      - The response headers are set.
    - When a preflight request is received, the `request.attributes.isCORSPreflight` flag is set to `true`.
    - For all preflight requests, the `security` and `client-identification` preflow policies are always skipped, whether CORS is enabled or not enabled.
7. Optional: Select Activate API  if you want to immediately use the API for further development and testing.

Note:

- When you select the Activate API option, API Connect automatically completes the following actions:
    - Creates a draft Product, adds the API to the Product, and publishes the Product to the Sandbox Catalog so that the API is available to be called. The Product has the title *api\_title* auto product. This Product is not visible in the Develop view and you cannot delete it directly. However, if you delete the API the draft Product is deleted together with the API; see [Deleting an API definition](#). The Product is visible in any Catalogs to which it is published. If you want to remove the Product from a Catalog, you must do this separately; see [Removing a Product from a Catalog](#)
    - Subscribes the Sandbox test application to the Product so that you can immediately test the API in the test environment. For information on testing an API, see [Testing an API](#).
  - You cannot use the Activate API option if lifecycle approval is enabled in the Sandbox Catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to use the Activate API option they must be disabled; for information on lifecycle approval settings, see [Creating and configuring Catalogs](#).
  - To use the Activate API option, you must be assigned a role that has the **Product:Manage** and **Subscription:Manage** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see [Creating custom roles](#).
8. Click Next to create your API definition.  
The Summary panel displays messages as the definition is created, and the selected security options and rate limits are enforced.
- If you selected Activate API, the wizard populates an API Endpoint URL that you can use in testing. If you have also selected Secure using Client ID, the wizard displays a Client ID and Client Secret you can use.
9. Select one of the following options:
- To further configure your API, click Edit API. For details, see [Editing an API definition](#).
  - If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see [Editing an API definition](#).

## Results

---

You created a proxy API from an existing OpenAPI service.

## What to do next

---

APIs are made available to application developers by including them in a Product, and then publishing that Product to a Catalog. For more information, see [Working with Products](#) and [Working with Catalogs](#).

## Related tasks

---

- [Defining Paths for a REST API](#)

## Related reference

---

- [API and Product definition template examples](#)

# Creating a REST proxy API from an existing WSDL service

If you have a SOAP service defined in a WSDL file, you can use the WSDL file to create an API Connect REST proxy API that calls that SOAP service.

## About this task

---

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To complete this task, you must be assigned a role that has the **Api-Drafts:Edit**, **Settings:View**, and **App:View** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see [Creating custom roles](#).


If the WSDL file is a stand-alone file with no external dependencies, you can load the .wsdl file from a directory to create the REST API definition.

If the WSDL file references other WSDL files or references XSD files containing XML schema definitions, you must create a .zip archive of the WSDL file and its dependent documents, and then load the .zip file from a directory to add the REST API definition.

## Procedure

---

To add a REST API definition by loading a WSDL file, complete the following steps:

1. In the navigation pane, click  Develop, select the APIs tab, then click Add API (from REST, GraphQL or SOAP).  
The Select API type screen is displayed.
2. Select OpenAPI 2.0; this API type does not support OpenAPI 3.0.
3. On the Add API pane, select From existing WSDL service (REST Proxy). Click Next.  
The Create API from Existing WSDL Service (REST Proxy) window opens.
4. To upload the service information from a stand-alone .wsdl file, or a .zip file that contains a WSDL file and its dependent documents, you can either drag and drop your file, or browse and select the file that you want to use.  
After you upload the file, the WSDL is evaluated and a message displays whether the WSDL is valid.  
  
If you upload a .zip file, you can include in the .zip file an options file to specify additional directives. For details, see [Using an options file when importing a WSDL service](#).
5. Click Next. At the Select Service panel, select a WSDL service from the imported file.

6. Click Next. Specify the API summary in the Info section. You can fine-tune the API after it is created.
  - The Title can include special characters but should be kept short so that it can be easily displayed in the user interface.
  - The Name is entered automatically. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see [apic draft-apis](#).
  - The Version corresponds to the value of the `info.version` property of the API's OpenAPI definition. The `version.release.modification` version numbering scheme is recommended; for example `1.0.0`.
  - The Base path is the URL segment of the API and does not include the host name or any additional segments for paths or operations. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
  - The optional Description helps to identify the API.
7. Click Next. In the Secure section, configure the API security that you require.
  - Secure using Client ID - Select this option to require an Application to provide a Client ID (API Key). This causes the `X-IBM-Client-Id` parameter to be included in the request header for the API. If selected, you can then select whether to limit the API calls on a per key (per Client ID) basis:
    - Limit API calls on a per key basis - If selected, you must configure the rate limit that you require. Rate limits control the maximum number of calls allowed within a time period (hour, minute, month or day). For example, 100 calls per hour.

For information about security options in IBM® API Connect, see [Configuring API security](#).

- CORS - Select this option to enable cross-origin resource sharing (CORS) support for your API. This allows your API to be accessed from another domain.  
Note:
  - When CORS is enabled, the API Gateway runs the `cors` preflight policy to handle all CORS requests that are made to the API.
  - When CORS is enabled and a preflight request is received, only the following API actions are performed:
    - The `cors` preflight policy configures the appropriate response headers.
    - The response headers are set.
  - When a preflight request is received, the `request.attributes.isCORSPreflight` flag is set to `true`.
  - For all preflight requests, the `security` and `client-identification` preflight policies are always skipped, whether CORS is enabled or not enabled.
8. Optional: Select Activate API if you want to immediately use the API for further development and testing.  
Note:
  - When you select the Activate API option, API Connect automatically completes the following actions:
    - Creates a draft Product, adds the API to the Product, and publishes the Product to the Sandbox Catalog so that the API is available to be called. The Product has the title `api_title` auto product. This Product is not visible in the Develop view and you cannot delete it directly. However, if you delete the API the draft Product is deleted together with the API; see [Deleting an API definition](#). The Product is visible in any Catalogs to which it is published. If you want to remove the Product from a Catalog, you must do this separately; see [Removing a Product from a Catalog](#)
    - Subscribes the Sandbox test application to the Product so that you can immediately test the API in the test environment. For information on testing an API, see [Testing an API](#).
  - You cannot use the Activate API option if lifecycle approval is enabled in the Sandbox Catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to use the Activate API option they must be disabled; for information on lifecycle approval settings, see [Creating and configuring Catalogs](#).
  - To use the Activate API option, you must be assigned a role that has the `Product:Manage` and `Subscription:Manage` permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see [Creating custom roles](#).
9. Click Next to create your API definition.  
The Summary panel displays messages as the definition is created, and the selected security options and rate limits are enforced.  
  
If you selected Activate API, the wizard populates an API Endpoint URL that you can use in testing. If you have also selected Secure using Client ID, the wizard displays a Client ID and Client Secret you can use.
10. Select one of the following options:
  - To further configure your API, click Edit API. For details, see [Editing an API definition](#).
  - If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see [Editing an API definition](#).

---


## Results

You created a REST API definition by using an existing WSDL file.

---

## What to do next

APIs are made available to application developers by including them in a Product, and then publishing that Product to a Catalog. For more information, see [Working with Products](#) and [Working with Catalogs](#).

You can also manage the Product lifecycle, and control who can see and subscribe to your Product, by opening the Sandbox Catalog associated with your API in the  Manage page of the API Designer UI.

---

## Related concepts

- [Developing your APIs and applications](#)

---

## Related reference

- [API and Product definition template examples](#)

---

## Related information

- [IBM API Connect overview](#)

---

# Creating a SOAP proxy API from an existing WSDL service

If you have a SOAP service defined in a WSDL file, you can use the WSDL file to create an API Connect SOAP proxy API that calls that SOAP service.

## About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To complete this task, you must be assigned a role that has the **Api-Drafts:Edit**, **Settings:View**, and **App:View** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see [Creating custom roles](#).


You can add a SOAP API to expose an existing SOAP service by supplying the WSDL file that defines that existing service in one of the following ways:

- If the WSDL file is a stand-alone file with no external dependencies, you can load the .wsdl file from a directory to create the SOAP API definition.
- You can provide one or more WSDL files in a single .zip file that contains the WSDL files and any necessary schemas. If the WSDL file references other WSDL files or references XSD files containing XML schema definitions, you must create a .zip archive of the WSDL file and its dependent documents, and then load the .zip file from a directory to add the SOAP API definition.
- If you are using API Designer with existing APIs that were created from a WSDL or WSDL.zip file in Version 5 API Designer, make sure that the following requirements are met:
  - The WSDL or WSDL.zip file must be stored in the same directory as the API's YAML file.
  - In the API's YAML file, the **x-ibm-configuration --> wsdl-definition --> wsdl** property must point directly to the WSDL or WSDL.zip file with no path specified (the files are located in the same directory).

The service must support Web Services Basic Profile Version 1.1 - Second Edition.

## Procedure

To add a SOAP API definition by loading a WSDL file, complete the following steps:

1. In the navigation pane, click  Develop, select the APIs tab, then click Add API (from REST, GraphQL or SOAP). The Select API type screen is displayed.
2. Select OpenAPI 2.0; this API type does not support OpenAPI 3.0.
3. Select From existing WSDL service (SOAP Proxy). Click Next. The Create API from Existing WSDL Service (SOAP Proxy) window opens.
4. To upload the service information from a stand-alone .wsdl file, or a .zip file that contains a WSDL file and its dependent documents, you can either drag and drop your file, or browse and select the file that you want to use. After you upload the file, the WSDL is evaluated and a message displays whether the WSDL is valid. If you upload a .zip file, you can include in the .zip file an options file to specify additional directives. For details, see [Using an options file when importing a WSDL service](#).
5. Click Next. At the Select Service panel, select a WSDL service from the imported file.
6. Click Next. Specify the API summary in the Info section. You can fine-tune the API after it is created.
  - The Title can include special characters but should be kept short so that it can be easily displayed in the user interface.
  - The Name is entered automatically. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see [apic draft-apis](#).
  - The Version corresponds to the value of the **info.version** property of the API's OpenAPI definition. The **version.release.modification** version numbering scheme is recommended; for example 1.0.0.
  - The Base path is the URL segment of the API and does not include the host name or any additional segments for paths or operations. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
  - The optional Description helps to identify the API.
7. Click Next. In the Secure section, configure the API security that you require.
  - Secure using Client ID - Select this option to require an Application to provide a Client ID (API Key). This causes the **x-IBM-Client-Id** parameter to be included in the request header for the API. If selected, you can then select whether to limit the API calls on a per key (per Client ID) basis:
    - Limit API calls on a per key basis - If selected, you must configure the rate limit that you require. Rate limits control the maximum number of calls allowed within a time period (hour, minute, month or day). For example, 100 calls per hour.For information about security options in IBM® API Connect, see [Configuring API security](#).
  - CORS - Select this option to enable cross-origin resource sharing (CORS) support for your API. This allows your API to be accessed from another domain. Note:
    - When CORS is enabled, the API Gateway runs the **cors** preflow policy to handle all CORS requests that are made to the API.
    - When CORS is enabled and a preflight request is received, only the following API actions are performed:
      - The **cors** preflow policy configures the appropriate response headers.
      - The response headers are set.
    - When a preflight request is received, the **request.attributes.isCORSPreflight** flag is set to **true**.
    - For all preflight requests, the **security** and **client-identification** preflow policies are always skipped, whether CORS is enabled or not enabled.
8. Optional: Select Activate API if you want to immediately use the API for further development and testing. Note:
  - When you select the Activate API option, API Connect automatically completes the following actions:
    - Creates a draft Product, adds the API to the Product, and publishes the Product to the Sandbox Catalog so that the API is available to be called. The Product has the title *api\_title* auto product. This Product is not visible in the Develop view and you cannot delete it directly. However, if you delete the API the draft Product is deleted together with the API; see [Deleting an API definition](#). The Product is visible in any Catalogs to which it is published. If you want to remove the Product from a Catalog, you must do this separately; see [Removing a Product from a Catalog](#)
    - Subscribes the Sandbox test application to the Product so that you can immediately test the API in the test environment. For information on testing an API, see [Testing an API](#).
  - You cannot use the Activate API option if lifecycle approval is enabled in the Sandbox Catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to use the Activate API option they must be disabled; for information on lifecycle approval settings, see [Creating and configuring Catalogs](#).
  - To use the Activate API option, you must be assigned a role that has the **Product:Manage** and **Subscription:Manage** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see [Creating custom roles](#).

9. Click Next to create your API definition.

The Summary panel displays messages as the definition is created, and the selected security options and rate limits are enforced.

If you selected Activate API, the wizard populates an API Endpoint URL that you can use in testing. If you have also selected Secure using Client ID, the wizard displays a Client ID and Client Secret you can use.

10. Select one of the following options:

- To further configure your API, click Edit API. For details, see [Editing an API definition](#).
- If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see [Editing an API definition](#).

## Results


---

You created a SOAP API definition by using an existing WSDL file.

## What to do next

---

APIs are made available to application developers by including them in a Product, and then publishing that Product to a Catalog. For more information, see [Working with Products](#) and [Working with Catalogs](#).

You can also manage the Product lifecycle, and control who can see and subscribe to your Product, by opening the Sandbox Catalog associated with your API in the  Manage page of the API Designer UI.

## Related concepts

---

- [Developing your APIs and applications](#)

## Related reference

---

- [API and Product definition template examples](#)

## Related information

---

- [IBM API Connect overview](#)

# Creating a new REST OpenAPI definition

You can create and edit draft REST API definitions by using the API Designer or the API Designer user interface in IBM® API Connect.

## About this task

---


You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

**API Manager UI only:** To complete this task you must be assigned a role that has the **Api-Drafts:Edit** permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see [Creating custom roles](#).

## Procedure

---

To create a new REST OpenAPI definition, complete the following steps.

1. In the navigation pane, click  Develop, select the APIs tab, then click Add API (from REST, GraphQL or SOAP).  
The Select API type screen is displayed.
2. Select OpenAPI 2.0 or OpenAPI 3.0 according to which version of the OpenAPI specification your API is to be based on.
3. Select New OpenAPI.
4. Click Next. Specify the API summary in the Info section. You can fine-tune the API after it is created.
  - The Title can include special characters but should be kept short so that it can be easily displayed in the user interface.
  - The Name is entered automatically. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see [apic draft-apis](#).
  - The Version corresponds to the value of the **info.version** property of the API's OpenAPI definition. The **version.release.modification** version numbering scheme is recommended; for example 1.0.0.
  - The Base path is the URL segment of the API and does not include the host name or any additional segments for paths or operations. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
  - The optional Description helps to identify the API.
5. Click Next. In the Secure section, configure the API security that you require.
  - Secure using Client ID - Select this option to require an Application to provide a Client ID (API Key). This causes the **X-IBM-Client-Id** parameter to be included in the request header for the API. If selected, you can then select whether to limit the API calls on a per key (per Client ID) basis:
    - Limit API calls on a per key basis - If selected, you must configure the rate limit that you require. Rate limits control the maximum number of calls allowed within a time period (hour, minute, month or day). For example, 100 calls per hour.For information about security options in IBM API Connect, see [Configuring API security](#).
  - CORS - Select this option to enable cross-origin resource sharing (CORS) support for your API. This allows your API to be accessed from another domain.  
Note:
    - When CORS is enabled, the API Gateway runs the **cors** preflow policy to handle all CORS requests that are made to the API.
    - When CORS is enabled and a preflight request is received, only the following API actions are performed:

- The `cors` preflight policy configures the appropriate response headers.
  - The response headers are set.
  - When a preflight request is received, the `request.attributes.isCORSPreflight` flag is set to `true`.
  - For all preflight requests, the `security` and `client-identification` preflight policies are always skipped, whether CORS is enabled or not enabled.
6. Click Next to create your API definition.  
The Summary panel displays messages as the definition is created, and the selected security options and rate limits are enforced.
7. Select one of the following options:
- To further configure your API, click Edit API. For details, see [Editing an API definition](#).
  - If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see [Editing an API definition](#).

## Results

You successfully created a REST API definition. For API Designer, the specifications for the APIs and Products are stored in the directory that you specified when you logged in. For API Manager, the specifications for the APIs and Products are stored on the management server.

## What to do next

APIs are made available to application developers by including them in a Product, and then publishing that Product to a Catalog. For more information, see [Working with Products](#) and [Working with Catalogs](#).

## Related tasks

- [Defining Paths for a REST API](#)
- [Configuring API security \(OpenAPI 2.0\)](#)
- [Configuring API security \(OpenAPI 3.0\)](#)

## Related reference

- [API and Product definition template examples](#)

# Adding a REST API by importing an OpenAPI definition file

You can use an OpenAPI definition file to import a REST API into IBM® API Connect.

## Before you begin

Your file must conform to version 2.0 or version 3.0 of the OpenAPI specification. The format of the file can be JSON or YAML.

Note: Products that contain an API with a Swagger property using `regex` that include lookahead assertions, such as "(?)" cannot be validated or published. An error message is returned. For example:

```
Product has not been published!
The multipart 'openapi' field contains an OpenAPI definition with validation errors.
 definitions.properties.pattern Does not match format 'regex' (context: (root).definitions.properties.pattern, line: 0, col:
0)
400
```

## About this task


You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

**API Manager UI only:** To complete this task, you must be assigned a role that has the `Api-Drafts:Edit`, `Settings:View`, and `App:View` permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see [Creating custom roles](#).

If you are using the API Designer UI and you want to use an OpenAPI definition file from elsewhere, downloaded from an external website for example, rather than created by using API Connect, use the import mechanism described here rather than copying the file into your local API Designer directory; the import operation adds API Connect specific sections that are required by API Designer.

## Procedure

To add a REST API by importing an OpenAPI file, complete the following steps:

1. In the navigation pane, click  Develop, select the APIs tab, then click Add API (from REST, GraphQL or SOAP).  
The Select API type screen is displayed.
2. Select OpenAPI 2.0 or OpenAPI 3.0 according to which version of the OpenAPI specification your API is to be based on.
3. In the Import section, select Existing OpenAPI, then click Next.
4. Choose one of the following methods for importing the file:
  - Drag and drop a file  
Open another window, select a file. Drag the file to the Select a file box, and release the file.
  - Browse for a file

Click the Select a file box. Locate the file and select it.

- Specify the URL of a file  
In the Or specify a file URL field, type or paste a URL for the file. Optionally add a User name and Password if they are required for accessing the file.

The following file types are supported if they contain a valid OpenAPI definition: .json, .yaml, and .yml.

5. Click Next to import the selected file.

The wizard checks the validity of the YAML, and displays a message indicating successful validation.

Notes:

- If you are using the API Manager UI, the import operation fails if the file defines an API that has the same name and version as an existing API definition. However, if you are using the API Designer UI, API definitions are uniquely identified by the file name in your local file system; therefore, if you import two different files that define the same API name and version, two API definitions with the same name and version are created in API Connect, with no error.
- Any validation error messages are displayed in English only, and are not translated.

6. Optional: Select Activate API if you want to immediately use the API for further development and testing.

Note:

- When you select the Activate API option, API Connect automatically completes the following actions:
  - Creates a draft Product, adds the API to the Product, and publishes the Product to the Sandbox Catalog so that the API is available to be called. The Product has the title *api\_title* auto product. This Product is not visible in the Develop view and you cannot delete it directly. However, if you delete the API the draft Product is deleted together with the API; see [Deleting an API definition](#). The Product is visible in any Catalogs to which it is published. If you want to remove the Product from a Catalog, you must do this separately; see [Removing a Product from a Catalog](#)
  - Subscribes the Sandbox test application to the Product so that you can immediately test the API in the test environment. For information on testing an API, see [Testing an API](#).
- You cannot use the Activate API option if lifecycle approval is enabled in the Sandbox Catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to use the Activate API option they must be disabled; for information on lifecycle approval settings, see [Creating and configuring Catalogs](#).
- To use the Activate API option, you must be assigned a role that has the **Product:Manage** and **Subscription:Manage** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see [Creating custom roles](#).

7. Click Next. The Import API Summary panel indicates that the YAML file is loaded and valid.

If you selected Activate API, the wizard populates an API Endpoint URL, and displays a Client ID and Client Secret that you can use.

8. Select one of the following options:

- To further configure your API, click Edit API. For details, see [Editing an API definition](#).
- If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see [Editing an API definition](#).

## Results

---

When the API definition has been imported, it is shown in the list of API definitions in the Develop page. For API Designer, the specifications for the APIs and Products are stored in the directory that you specified when you logged in. For API Manager, the specifications for the APIs and Products are stored on the management server.

## What to do next

---

APIs are made available to application developers by including them in a Product, and then publishing that Product to a Catalog. For more information, see [Working with Products](#) and [Working with Catalogs](#).

## Related information

---

- [IBM API Connect overview](#)
- [Managing your APIs](#)
- [OpenAPI Specification](#)
- [What is OpenAPI?](#)
- [API and Product definition template examples](#)

## Adding a SOAP API by importing a zip file

---

You can use an OpenAPI definition file to import a SOAP API into IBM® API Connect.

## Before you begin

---

The format of the file must be .zip, and the file must contain the WSDL definition and the YAML file that defines the API. The YAML file must conform to version 2.0 of the OpenAPI specification.

Note: Products that contain an API with a Swagger property using **regex** that include lookahead assertions, such as "(?" cannot be validated or published. An error message is returned. For example:

```
Product has not been published!
The multipart 'openapi' field contains an OpenAPI definition with validation errors.
definitions.properties.pattern Does not match format 'regex' (context: (root).definitions.properties.pattern, line: 0, col:
0)
400
```

## About this task

---

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

**API Manager UI only:** To complete this task, you must be assigned a role that has the **Api-Drafts:Edit**, **Settings:View**, and **App:View** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see [Creating custom roles](#).


If you are using the API Designer UI and you want to use an OpenAPI definition file from elsewhere, downloaded from an external website for example, rather than created by using API Connect, use the import mechanism that is described here rather than copying the file into your local API Designer directory; the import operation adds API Connect specific sections that are required by API Designer.

For more information on how to create a .zip file, see [Downloading an API definition](#).

## Procedure

---

To add a SOAP API by importing a .zip file, complete the following steps:

1. In the navigation pane, click  Develop, select the APIs tab, then click Add API (from REST, GraphQL or SOAP).  
The Select API type screen is displayed.
2. Select OpenAPI 2.0; this API type does not support OpenAPI 3.0.
3. In the Import section, select Existing OpenAPI, then click Next.
4. To import a file from your local file system, you can either drag and drop the file or click the link to select the file from your local file system.  
The wizard checks the validity of the WSDL and YAML, and displays a message that indicates a successful validation.  
Note:
  - If you are using the API Manager UI, the import operation fails if the file defines an API that has the same name and version as an existing API definition. However, if you are using the API Designer UI, API definitions are uniquely identified by the file name in your local file system; therefore, if you import two different files that define the same API name and version, two API definitions with the same name and version are created in API Connect, with no error.
  - Any validation error messages are displayed in English only, and are not translated.
5. Click Next.
6. Optional: Select Activate API if you want to immediately use the API for further development and testing.  
Note:
  - When you select the Activate API option, API Connect automatically completes the following actions:
    - Creates a draft Product, adds the API to the Product, and publishes the Product to the Sandbox Catalog so that the API is available to be called. The Product has the title *api\_title* auto product. This Product is not visible in the Develop view and you cannot delete it directly. However, if you delete the API the draft Product is deleted together with the API; see [Deleting an API definition](#). The Product is visible in any Catalogs to which it is published. If you want to remove the Product from a Catalog, you must do this separately; see [Removing a Product from a Catalog](#)
    - Subscribes the Sandbox test application to the Product so that you can immediately test the API in the test environment. For information on testing an API, see [Testing an API](#).
  - You cannot use the Activate API option if lifecycle approval is enabled in the Sandbox Catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to use the Activate API option they must be disabled; for information on lifecycle approval settings, see [Creating and configuring Catalogs](#).
  - To use the Activate API option, you must be assigned a role that has the **Product:Manage** and **Subscription:Manage** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see [Creating custom roles](#).
7. Click Next. The Import API Summary panel indicates that the file is loaded and valid.  
If you selected Activate API, the wizard populates an API Endpoint URL, and displays a Client ID and Client Secret that you can use.
8. Select one of the following options:
  - To further configure your API, click Edit API. For details, see [Editing an API definition](#).
  - If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see [Editing an API definition](#).

## Results

---

When the API definition has been imported, it is shown in the list of API definitions in the Develop page. For API Designer, the specifications for the APIs and Products are stored in the directory that you specified when you logged in. For API Manager, the specifications for the APIs and Products are stored on the management server.

## What to do next

---

APIs are made available to application developers by including them in a Product, and then publishing that Product to a Catalog. For more information, see [Working with Products](#) and [Working with Catalogs](#).

## Related information

---

- [IBM API Connect overview](#)
- [Managing your APIs](#)
- [OpenAPI Specification](#)
- [What is OpenAPI?](#)
- [API and Product definition template examples](#)

API Gateway only

## Creating a GraphQL proxy API

---

GraphQL is a query language for APIs that gives an application client greater control over what data it retrieves in an API request when compared with a REST API request. IBM® API Connect enables you to create a GraphQL API proxy definition that proxies a backend GraphQL server, and to define rate limiting controls that reflect the amount of data that is returned from the server by a request to the GraphQL API.



## About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To complete this task, you must be assigned a role that has the **Api-Drafts:Edit**, **Settings:View**, and **App:View** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see [Creating custom roles](#).


GraphQL provides the following particular advantages over REST APIs:

- The application client can request only the data that it needs. For example, when retrieving bank account records, request only the account number and current balance for each account, but not the customer name and address details. With a REST API request, either the backend REST service must provide separate endpoints for different data subsets, or the application client must retrieve the complete records and then discard the unwanted data.
- The application client can retrieve multiple related resources in a single request. For example, a customer's bank account record might include an array that references other finance products that the customer holds. If an application client wants to retrieve the bank account details for a specific customer, and details of the other finance products for that customer, then with a REST API the client would first retrieve the bank account details, then make separate requests for each of the other products. A GraphQL API can be designed to allow the client to retrieve all this information in a single request.

However, this flexibility presents rate limiting challenges, because two seemingly very similar requests might return vastly different amounts of data, and what might have required multiple REST API requests, each counting towards the rate limit, might require only a single GraphQL API request. It is important therefore that rate limiting controls are imposed that reflect the amount of data that is retrieved. API Connect extends the GraphQL standard by providing, in a GraphQL API definition, the ability to configure a range of settings that are used to calculate the complexity of a GraphQL request and an associated cost that counts towards the rate limit.

## Procedure

To create a GraphQL API proxy definition, complete the following steps:

1. In the navigation pane, click  Develop, select the APIs tab, then click Add API (from REST, GraphQL or SOAP).  
The Select API type screen is displayed.
2. Select OpenAPI 2.0; this API type does not support OpenAPI 3.0.
3. Select From existing GraphQL service (GraphQL proxy).
4. Click Next. Specify the API summary in the Info section. You can fine-tune the API after it is created.
  - The Title can include special characters but should be kept short so that it can be easily displayed in the user interface.
  - The Name is entered automatically. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see [apic draft-apis](#).
  - The Version corresponds to the value of the `info.version` property of the API's OpenAPI definition. The `version.release.modification` version numbering scheme is recommended; for example `1.0.0`.
  - The Base path is the URL segment of the API and does not include the host name or any additional segments for paths or operations. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
  - The optional Description helps to identify the API.
5. In the GraphQL Server section, specify the following values:
  - GraphQL server URL: this is the URL that is used for requests to the backend GraphQL API. The GraphQL schema is imported automatically from the specified URL.
  - Schema name: the name that you enter here is used to identify the schema definition that is created from the imported GraphQL schema. The field is pre-populated with host name segment of the value entered in the GraphQL server URL field, but you can change it.
6. Click Next. If there are any warnings indicated relating to the imported schema, you can review them now by clicking View.  
You can deal with these warnings as appropriate after you have created your GraphQL proxy API.
7. In the Operations and Paths section, select the operations and paths that you want to include.  
The operation path of `/graphql` is already selected and you cannot change this setting; this is the path for application client requests to API Connect that invoke the GraphQL API.

For details of the mechanisms supported for sending a request to the `/graphql/cost` endpoint, see [Request mechanisms supported by GraphQL endpoints](#).

You can also select the operation path of `graphql/cost`; this path enables application clients to obtain details of the cost of a request to the GraphQL API before making the actual request. In a production environment, in particular, you should consider carefully the resource implications of making this path available. For full details regarding the enablement of the cost endpoint, see [Enabling the cost endpoint for a GraphQL API](#).

To form the full API paths, these path segments are appended to the `basepath` that you specified in step 4.

To allow application clients to send introspection requests for this GraphQL proxy API, select Support default introspection. For full details, see [Supporting introspection for a GraphQL API](#).

To allow users to test the GraphQL proxy API from a GraphQL editor, select Enable GraphQL editor. For full details, see [Enabling the GraphQL editor for a GraphQL API](#).

Note: The `graphql/cost`, Support default introspection, and Enable GraphQL editor options each automatically generate policy configuration in your API assembly that would have to be created manually if you disable this option and later decide that it is required.

8. Optionally, you can replace the GraphQL schema that was imported from the GraphQL server URL by uploading a schema from your local file system. Click Replace; you can then either drag and drop your file, or click where indicated to select the file from your local file system. Click Upload when done.  
Note:
  - The schema must be written in the GraphQL Schema Definition Language (SDL). For more information, see [Type language](https://graphql.org/) in the GraphQL documentation at <https://graphql.org/>.
  - If you upload a schema from a local file having previously uploaded a schema from a URL, the schema URL setting that is stored in the OpenAPI source for the API, and subsequently displayed in the user interface, retains the previously specified URL value rather than indicating a file location.
9. Click Next. In the Secure section, configure the API security that you require.
  - Secure using Client ID - Select this option to require an Application to provide a Client ID (API Key). This causes the `X-IBM-Client-Id` parameter to be included in the request header for the API. For information about security options in IBM API Connect, see [Configuring API security](#).
  - CORS - Select this option to enable cross-origin resource sharing (CORS) support for your API. This allows your API to be accessed from another domain.  
Note:
    - When CORS is enabled, the API Gateway runs the `cors` preflow policy to handle all CORS requests that are made to the API.
    - When CORS is enabled and a preflight request is received, only the following API actions are performed:
      - The `cors` preflow policy configures the appropriate response headers.

- The response headers are set.
  - When a preflight request is received, the `request.attributes.isCORSPreflight` flag is set to `true`.
  - For all preflight requests, the `security` and `client-identification` preflow policies are always skipped, whether CORS is enabled or not enabled.
10. Optional: Select Activate API if you want to immediately use the API for further development and testing.
- Note:
- When you select the Activate API option, API Connect automatically completes the following actions:
    - Creates a draft Product, adds the API to the Product, and publishes the Product to the Sandbox Catalog so that the API is available to be called. The Product has the title `api_title` auto product. This Product is not visible in the Develop view and you cannot delete it directly. However, if you delete the API the draft Product is deleted together with the API; see [Deleting an API definition](#). The Product is visible in any Catalogs to which it is published. If you want to remove the Product from a Catalog, you must do this separately; see [Removing a Product from a Catalog](#)
    - Subscribes the Sandbox test application to the Product so that you can immediately test the API in the test environment. For information on testing an API, see [Testing an API](#).
  - You cannot use the Activate API option if lifecycle approval is enabled in the Sandbox Catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to use the Activate API option they must be disabled; for information on lifecycle approval settings, see [Creating and configuring Catalogs](#).
  - To use the Activate API option, you must be assigned a role that has the `Product:Manage` and `Subscription:Manage` permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see [Creating custom roles](#).
11. Click Next to create your API definition.  
The Summary panel displays messages as the definition is created, and the selected security options are enforced.
- If you selected Activate API, the wizard populates an API Endpoint URL that you can use in testing. If you have also selected Secure using Client ID, the wizard displays a Client ID and Client Secret you can use.
12. Click Next to create your API definition.  
The Summary panel displays messages as the definition is created, and the selected security options and rate limits are enforced.
13. Select one of the following options:
- To further configure your API, click Edit API. For details, see [Editing an API definition](#).
  - If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see [Editing an API definition](#).

## What to do next

Use the GraphQL schema editor to review, and deal with, any warnings associated with the schema, and to configure a range of settings that are used to calculate the complexity of a GraphQL request, and an associated cost that counts towards the rate limit; for details, see [Using the GraphQL schema editor](#).

- [Using the GraphQL schema editor](#)  
The GraphQL schema editor allows you to a configure a range of settings that are used to calculate the complexity of a GraphQL request, and an associated cost that counts towards the rate limit.
- [Supporting introspection for a GraphQL API](#)  
If you select the Support default introspection option when creating a GraphQL proxy API, client applications can send GraphQL introspection requests to the API.
- [Enabling the GraphiQL editor for a GraphQL API](#)  
If you select the Enable GraphiQL editor option when creating a GraphQL proxy API, users can test the API by using a GraphiQL editor in a web browser; API Connect responds to the initial GraphiQL editor request with an HTML page that enables testing from the GraphiQL editor.
- [Enabling the cost endpoint for a GraphQL API](#)  
If you enable the `/graphql/cost` endpoint when creating a GraphQL proxy API, a calling application can obtain details of the cost of executing a request on the backend GraphQL server before making the actual backend request.
- [Securing a GraphQL API by using a client ID](#)  
If you select the Secure using Client ID option when creating a GraphQL proxy API, API Connect automatically adds Rate limit policies to the API assembly.
- [Request mechanisms supported by GraphQL endpoints](#)  
API Connect supports various mechanisms for sending requests to GraphQL proxy API endpoints.
- [GraphQL custom schema directives and scalar types](#)  
The API Connect DataPower® API Gateway extends the GraphQL specification by providing additional schema directives and custom scalar types.
- [@remove directive](#)  
The `@remove` GraphQL schema directive specifies conditions for removing types and fields from validation or introspection for each transaction based on values in the API context.
- [GraphQL limitations](#)  
This page details the limitations in the API Connect GraphQL implementation for the current release.

API Gateway only

## Using the GraphQL schema editor

The GraphQL schema editor allows you to a configure a range of settings that are used to calculate the complexity of a GraphQL request, and an associated cost that counts towards the rate limit.

### Introduction

When the GraphQL schema editor opens, all the types defined in the GraphQL schema are listed. You can expand a type to display all the fields that are defined for that type.

For each type, you can specify the following setting:

Type weight

The weighting factor that you want to apply to the type for use when calculating the type cost for a request to the GraphQL API. In general, the weighting factor for a type will reflect the resource implications for a request that retrieves values of that type; for example:

- A complex object might be weighted more heavily than a String.
- Values of one type might be retrieved from a different backend database to values of another type, and accessing one database requires more CPU resource, so types associated with this database are weighted more heavily.

For each field, you can specify the following settings:

#### Field weight

The weighting factor that you want to apply to the field for use when calculating the field cost of a request to the GraphQL API.

#### Assumed size

The size limit that the backend server imposes on the number of values of this field that are returned, in a list, for each separate retrieval of the field in an API request. If an assumed size has been defined for this field in the imported GraphQL schema, its value is displayed. You can enter a value, or change the current value, and the schema definition stored in API Connect is updated accordingly. You must ensure that the specified value corresponds with the implementation or configuration on the backend server so that API Connect can apply rate limiting correctly. The assumed size is intended to be configured for fields that return lists of values, to tell the cost analysis how many values to expect.



#### Slicing arguments

A comma separated list of the arguments that are used by the backend server to limit the number of values of this field that are returned, in a list, for each separate retrieval of the field in an API request. These must correspond with the implementation or configuration on the backend server. For example, there might be slicing arguments of `first` and `last` that are used to specify a range of objects to be returned by an API request.

#### Sized fields

In some cases, a field with a configured Assumed size or Slicing arguments setting returns an object, a field of which returns the values whose number is to be sized. This is common in the [connections pattern for pagination](#), where fields with slicing arguments `first` or `last` return a `connections` object, that contains the sized list in an `edges` field. The Sized fields option allows the cost analysis to consider these indirect relationships between assumed sizes or slicing arguments and the sized list of values. Supply a comma separated list of field names.

You access the GraphQL schema editor in either of the following ways:

- Immediately after creating a new GraphQL proxy API, by clicking Edit API, then clicking the GraphQL Schema tab. For GraphQL proxy API creation details, see [Creating a GraphQL proxy API](#).
- By modifying the configuration of an existing API; complete the following steps:
  1. In the navigation pane, click  Develop, then select the APIs tab.
  2. Alongside the GraphQL API definition that you want to work with, click the options icon  and then click Edit API.
  3. Click the GraphQL Schema tab.

## Modifying a setting for a single type or field

---

For a numerical setting, click the setting alongside the required type or field, then enter the required value, or use the stepper control. For a setting that has a textual value, enter the value directly. You might first want to filter the display to show only a relevant subset of types and fields; see [Filtering the display](#).

## Modifying settings for multiple types and fields

---

To modify settings for multiple types and fields in a single operation, complete the following steps:

1. Select the check boxes alongside the types and fields whose settings you want to modify.
2. Click Apply analysis configuration.
3. On the Type tab, specify the settings that you want to apply to the selected types.
4. On the Field tab, specify the settings that you want to apply to the selected fields.
5. Click Apply to apply the settings to the selected types and fields.
6. When you have made all the required modifications to the selected types and fields, click Done; this action clears all check box settings.

To restore any type or field settings to the initial default settings after having applied an analysis configuration, select the check boxes alongside the required types and fields, then click Remove analysis configuration.

You might first want to filter the display to show only a relevant subset of types and fields; see [Filtering the display](#).



## Filtering the display

---

To display only those type and fields that contain a specified string, enter the required filter string in the Search schema field, then press Enter.

If a type matches the filter, the type and all its fields are displayed. If a field matches the filter but not its parent type, the parent type is displayed together with all matching fields. You can enter multiple filter strings; the items displayed are those that contain all strings.


You can now use the operations described in [Modifying a setting for a single type or field](#) and [Modifying settings for multiple types and fields](#) to update the settings for the displayed types and fields.

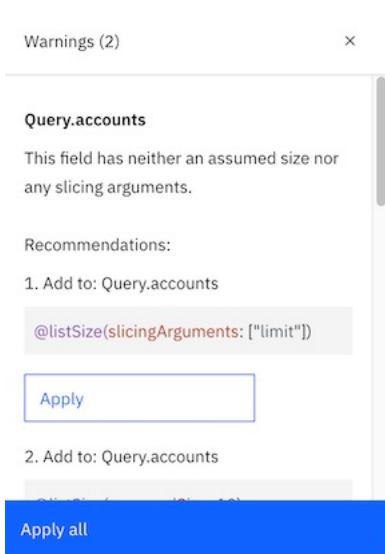
To remove a specific filter string from the search, click the Close icon  for that filter string. To restore the full type and field display by removing all filter strings, click the Clear icon  at the end of the search field.



## GraphQL schema warnings

---

Where applicable, API Connect will recommend updates to the analysis configuration to protect the backend server from excessive resource utilization, and to ensure correct cost estimation for requests. These recommendations are displayed as warnings in the GraphQL schema editor.

If a type contains one or more fields with warnings, this is indicated by a warning count icon  2 alongside the type. You can click the warning count icon to see the warnings for that type; the Warnings window opens at the relevant warnings, for example:



To see the fields that have warnings, expand the type; a warning icon  is displayed alongside fields with warnings. To see the details of a warning for a field, click the warning icon  alongside the field

If you click Accept under any warning in the Warnings window, the recommended update is applied to the schema automatically.

To apply the suggested updates for all warnings in a single operation, click Apply all in the Warnings window, then click Apply to confirm.


You can filter the display to include only items with warnings; click the Menu icon , then select Show warnings only.

## Applying @remove directives to types and fields in the GraphQL schema

The **@remove** GraphQL schema directive specifies conditions for removing items from validation or introspection for each transaction based on values in the API context. For example, you can prevent specific Plans from using certain types or fields.

For more details on the **@remove** directive, including examples, see [rapid\\_graphql\\_extensions.html#reference\\_wrt\\_bk1\\_5mb\\_at\\_remove](#).

To apply an **@remove** directive to an item in your GraphQL schema, complete the following steps:


1. Click the settings icon  in the Show/hide column alongside the required item. The Show/hide settings window opens.
2. To remove the item depending on a boolean condition, enter the boolean expression in the Custom expression field; the following directive will be added to the item in the schema:

```
@remove (if: [expression])
```

To remove the item unconditionally, change the Show in schema slider control to Hide in schema; the following directive will be added to the item in the schema:

```
@remove (if: [\"true\"])
```

3. Click Next. A summary window opens. If the removal is permitted, any related items that will also have the **@remove** directive applied are listed; for example, if you apply an **@remove** directive to a type, and that type is referenced as the data type of a field on another type, the **@remove** directive is also applied to that field. If the removal is not permitted, an explanation is displayed.
4. If the removal is permitted, click Done to apply the **@remove** directive. If the removal is not permitted, either click Cancel to close the window or click Back and amend your settings.
5. To see the updates to the GraphQL JSON schema, click View source.

You can use the settings icon  to modify or delete previously applied **@remove** directives. To remove a conditional **@remove** directive, delete the expression from the Custom expression field. To remove an unconditional **@remove** directive, change the Hide in schema slider control to Show in schema.

## Viewing the GraphQL schema

To view the GraphQL JSON schema that was previously uploaded, click View source. Note that the schema includes **directive** statements and attribute values, added by API Connect, that reflect the configuration settings.

## Replacing the GraphQL schema

To replace the GraphQL schema, complete the following steps:

1. Click Replace.
2. Specify the replacement schema:
  - To replace the schema by introspecting a URL, complete the following steps:
    - Select Introspect from URL.
    - Enter the URL in the GraphQL server URL field, then click Introspect.

Note: If you replace the schema from a URL that is different to the one specified as the backend GraphQL server, in the GraphQL server URL field, when the GraphQL proxy API was created, the schema itself is replaced but the backend URL remains unchanged. You must correspondingly modify the URL setting on the **invoke** policy in the API assembly of your GraphQL proxy API.
  - To replace the schema by uploading a file, complete the following steps:
    - Select Upload schema definition language (SDL) file.

- Either drag and drop the file or click the link to select the file from your local file system.

Note:

- The schema must be written in the GraphQL Schema Definition Language (SDL). For more information, see [Type language](#) in the GraphQL documentation at <https://graphql.org/>.
- If you upload a schema from a local file having previously uploaded a schema from a URL, the schema URL setting that is stored in the OpenAPI source for the API, and subsequently displayed in the user interface, retains the previously specified URL value rather than indicating a file location.

All the changes between the replacement schema and the current schema are listed.

3. Update the schema as required:

- To have all new types and fields shown by default, select New schema for the Update schema defaults giving preference to setting. You can then choose to hide specific items by selecting Hide in the Action column alongside an item. Hiding an item causes an @remove directive to be added to the item in the schema source, as detailed in [Applying @remove directives to types and fields in the GraphQL schema](#).
- To have all new types and fields hidden by default, select Old schema for the Update schema defaults giving preference to setting. You can then choose to show specific items by selecting Show in the Action column alongside an item.

Note: If the current schema contains items that are not in the replacement schema, these items are deleted. Such deletions are shown in the change listing with the Accept setting displayed in the Action column; you cannot change this setting.

4. Click Next. A change summary page is displayed. If any removals are not permitted, an explanation is displayed, and you cannot complete the schema replacement; either click Cancel to close the window or click Back and amend your settings..
5. Click Submit to complete the schema replacement.

## Downloading the GraphQL schema

To download the GraphQL schema to a local file, click the Download. Note that the downloaded schema includes all additional settings that reflect your configuration updates.

## Related reference

- [GraphQL custom schema directives and scalar types](#)

API Gateway only

## Supporting introspection for a GraphQL API

If you select the Support default introspection option when creating a GraphQL proxy API, client applications can send GraphQL introspection requests to the API.

Introspection allows client applications to request details of all the types and fields that are defined in a GraphQL schema. When you create a GraphQL proxy API, introspection requests sent to that API from client applications are handled by the gateway. The response reflects any modifications you have made to the schema by using the [GraphQL schema editor](#).

When you select the Support default introspection option, API Connect creates the introspection endpoint. A GraphQL editor always attempts to introspect a GraphQL endpoint, so disabling this option significantly reduces the test capability for GraphQL editor users. This option also automatically generates policy configuration in your API assembly that would have to be created manually if you disable this option and later decide that it is required; see the [API assembly policies for introspection support](#) section for details. Disabling this option also limits the capability of the Developer Portal to test the GraphQL proxy API.

On the other hand, providing introspection support can have resource implications, and you might choose to communicate the schema information in another way. However, note that the GraphQL editor does not allow custom introspection queries to populate the document explorer. For more information on introspection, see [Introspection](#) in the GraphQL documentation at <https://graphql.org/>.

## API assembly policies for introspection support

This section details the API assembly policy configuration that is generated automatically if the Support default introspection option is selected when a GraphQL proxy API is created. If the option is not selected at creation time, and you later decide that you want to enable it, you must add it manually to the API assembly.

A **case** is added to a **switch** policy with the following condition:

```
($operationPath() = '/graphql' and message.attributes.parse.GraphQLIntrospection = 'standard-introspection')
```

If the condition is true, the following policies are executed in sequence, with property settings as indicated:

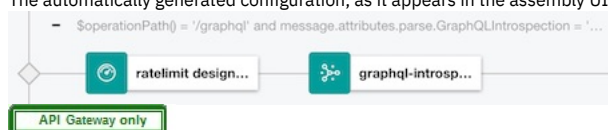
Rate limit

Property	Value
Source	Plan by Name
Rate Limit Name	graphql-design-request
Operation	Consume

GraphQL introspect

Property	Value
Input	message
Output	No value, defaults to input

The automatically generated configuration, as it appears in the assembly UI, is as follows:



## Enabling the GraphQL editor for a GraphQL API

If you select the Enable GraphQL editor option when creating a GraphQL proxy API, users can test the API by using a GraphQL editor in a web browser; API Connect responds to the initial GraphQL editor request with an HTML page that enables testing from the GraphQL editor.

Disabling the Enable GraphQL editor option reduces gateway traffic and latency and, furthermore, the Developer Portal provides equivalent test capability regardless of whether or not you enable the option.

This option automatically adds the following property setting to the OpenAPI source for the GraphQL proxy API:

```
html-page: 'store:///graphql.html'
```

If you disable the Enable GraphQL editor during API creation and later decide that it is required, you can either manually add this property setting, or you can select Enable GraphQL editor on the API Setup page of the Design tab of the API..

Note:

- The `html-page` property should be used **only** with a GraphQL proxy API.
- `store:///graphql.html` is the only value allowed for the `html-page` property.
- If you add the `html-page` property but the Support standard introspection property was not enabled during API creation, the GraphQL HTML page will not show any documentation regarding the GraphQL backend server, perform query validation, or perform smart lookup. For details on how to manually create the required assembly policy configuration, see [Supporting introspection for a GraphQL API](#).



## Enabling the cost endpoint for a GraphQL API

If you enable the `/graphql/cost` endpoint when creating a GraphQL proxy API, a calling application can obtain details of the cost of executing a request on the backend GraphQL server before making the actual backend request.

The "cost" does not relate purely to monetary cost but can take into account factors such as computational time, memory consumption, and other such factors, in addition to the actual financial costs associated with, say, using third-party systems to retrieve data.

In a production environment, in particular, you should consider carefully the resource implications of making this path available. Note, however, that enabling the `/graphql/cost` endpoint automatically generates policy configuration in your API assembly that would have to be created manually if you disable this option and later decide that it is required; see the [API assembly policies for the cost endpoint](#) section for details.

## API assembly policies for the cost endpoint

This section details the API assembly policy configuration that is generated automatically if the `/graphql/cost` operation path is selected when a GraphQL proxy API is created. If the path is not selected at creation time, and you later decide that you want to enable it, you must add it manually to the API assembly.

A `case` is added to a `switch` policy with the following condition:

```
($operationPath() = '/graphql/cost' and message.attributes.parse.GraphQLIntrospection = 'not-introspection')
```

If the condition is true, the following policies are executed in sequence, with property settings as indicated:

Rate limit

Property	Value
Source	Plan by Name
Rate Limit Name	graphql-design-request
Operation	Consume

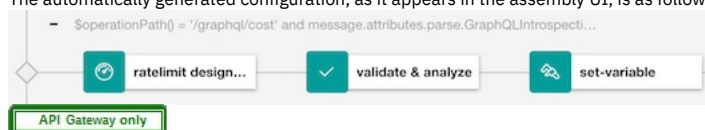
Validate

Property	Value
Input	message
Output	No value, defaults to input
Validate against	GraphQL

Set Variable

Property	Value
Action	Set
Set	message.body
Type	any
Value	\$(message.attributes.graphql)

The automatically generated configuration, as it appears in the assembly UI, is as follows:



## Securing a GraphQL API by using a client ID

If you select the Secure using Client ID option when creating a GraphQL proxy API, API Connect automatically adds Rate limit policies to the API assembly.

This section details the Rate limit policy configuration that is generated automatically. If the Secure using Client ID option is not selected at creation time, and you later decide that you want to enable it, you must add the appropriate Rate limit policies manually to the API assembly.

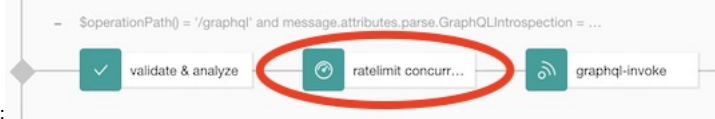
The automatically generated Rate limit policies are added to the **switch** cases in the assembly as follows:

- **switch case:**

```
($operationPath() = '/graphql' and message.attributes.parse.GraphQLIntrospection = 'not-introspection' and message.attributes.graphql.operationType = 'subscription')
```

A single Rate limit policy with the following property settings:

Property	Value
Source	Plan by Name
Count Limit Name	graphql-subscription-type-cost
Operation	Increment
Count Limit Name	graphql-subscription-field-cost
Operation	Increment



The policy as it appears in the assembly UI is as follows:

This Rate limit policy limits the number of HTML web browser requests for a GraphQL editor sent to the **/graphql** endpoint.

- **switch case:**

```
($operationPath() = '/graphql' and message.attributes.parse.GraphQLIntrospection = 'not-introspection' and message.attributes.graphql.operationType != 'subscription')
```

Two Rate limit policies with the following property settings:

- Policy 1:

Property	Value
Source	Plan by Name
Rate Limit Name	graphql-type-cost
Operation	Consume
Rate Limit Name	graphql-field-cost
Operation	Consume
Rate Limit Name	graphql-input-type-cost
Operation	Consume

This Rate limit policy limits calls to the GraphQL proxy API by calculating the expected cost of the query, and then correspondingly reducing the available cost balance that remains of the total configured cost limit. The expected cost is calculating by applying the weighting factors that have been attached to the types and fields in the GraphQL schema; for more information, see [Using the GraphQL schema editor](#).

The expected cost is calculated by the Validate policy that precedes the Rate limit policy and placed in the API context to be made available to the Rate limit policy. For details on the Validate policy, see [Validate](#).

The details of the cost calculations to which the referenced rate limits are applied are as follows:

- **graphql-field-cost:** The weighted sum of fields being selected in a query. Fields that return values of scalar or enum types have a weight of 0.0 and therefore don't affect field cost. For example, consider the following query, where it is assumed that the value **max** in the **filter** argument of field **Query.offices** constrains the number of **Office** objects being returned:

```
query {
 offices (filter: {location: "NY", max: 3}) {
 address {
 street
 }
 }
}
```

The **Address.street** field is of type **String** and therefore has a weight of 0.0. Non scalar and non enum field weights that are not changed in the GraphQL schema editor assume a default value of 1.0. If the **Office.address** field weight has been set to 3.0, the total field cost is 10.0, calculated as follows:

Field	Weight	Field cost
Query.offices	1.0	1 x 1.0 = 1.0
Office.address	2.0	3 x 3.0 = 9.0
Address.street	0.0	0.0
Total cost:		1.0 + 9.0 + 0.0 = 10.0

- **graphql-type-cost:** The weighted sum of types being returned by a query. Scalar or enum types have a weight of 0.0 and therefore don't affect type cost. Considering the previous example, and assuming that all types have a default weight of 1.0, the total type cost is 7 (1 x **Query** type + 3 x **Office** types + 3 x **Address** types).
- **graphql-input-type-cost:** The weighted sum of input types being sent in a query. Scalar or enum input types have a weight of 0.0 and therefore don't affect input type cost.

Considering the previous example, the input type cost is 1.0, because one `FilterInput` type is sent in the `filter` argument. `FilterInput.location` is a `String` type and `FilterInput.max` is an `Int` type, so they are not included as they are scalars.

- Policy 2:

Property	Value
Source	Plan by Name
Rate Limit Name	graphql-type-cost
Operation	Replenish
Rate Limit Name	graphql-field-cost
Operation	Replenish

This Rate limit policy limits calls to the GraphQL proxy API by calculating the actual cost of the query, and then correspondingly reducing the available cost balance that remains of the total configured cost limit. The actual cost is calculating by applying the weighting factors that have been attached to the types and fields in the GraphQL schema; for more information, see [Using the GraphQL schema editor](#).

Note that if the actual cost is less than the expected cost, this Rate limit policy will appropriately restore some of the available cost balance that had been removed by Policy 1, thereby overriding the original actual cost limitation. The actual cost might be less than the expected cost if, for example, the query requested more data than was available in a backend database. You can delete one or other of these Rate limit policies from the assembly if desired.

Note: Such unused cost is **not** restored if there are GraphQL response errors. The purpose of this restriction is to prevent malicious requests that cause large amounts of erroneous backend processing while only using a small amount of cost.

The actual cost is calculated by the Validate policy that precedes the Rate limit policy and placed in the API context to be made available to the Rate limit policy. For details on the Validate policy, see [Validate](#).

The policies as they appear in the assembly UI are as follows:



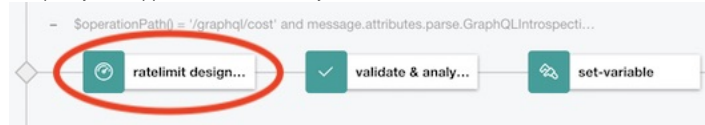
- switch case:

```
($operationPath() = '/graphql/cost' and message.attributes.parse.GraphQLIntrospection = 'not-introspection')
```

A single Rate limit policy with the following property settings:

Property	Value
Source	Plan by Name
Rate Limit Name	graphql-design-request
Operation	Consume

The policy as it appears in the assembly UI is as follows:



This Rate limit policy limits the requests sent to the `/graphql/cost` endpoint for retrieving the cost of a GraphQL query.

- switch case:

```
($operationPath() = '/graphql' and message.attributes.parse.GraphQLIntrospection = 'standard-introspection')
```

A single Rate limit policy with the following property settings:

Property	Value
Source	Plan by Name
Rate Limit Name	graphql-design-request
Operation	Consume

The policy as it appears in the assembly UI is as follows:



This Rate limit policy limits standard introspection requests sent to the `/graphql` endpoint.

The rate limits that are referred to by the Rate Limit Name settings in these policies are predefined in Plans that are created in Products that contain one or more GraphQL proxy APIs. You can configure the actual rate limiting values there; for more information, see [Editing a draft Product](#).

For more information on the Rate Limit policy, see [Rate Limit](#).



## Request mechanisms supported by GraphQL endpoints

API Connect supports various mechanisms for sending requests to GraphQL proxy API endpoints.

You can use any of the following mechanisms for sending requests to the `/graphql` and `/graphql/cost` endpoints on a GraphQL proxy API:



- **GET**: the GraphQL query must be specified in a URL query parameter called `query`.  
For example:

```
https://hostname/basepath/graphql?query={accounts(limit:100){name(first,last)}}
```

The following example includes the `variables` and `operationName` properties, with URL encoding:

```
https://hostname/basepath/graphql?
query=query%20fetchAccounts%20($!limit:%20Int)%20{accounts(limit:%20$!limit)%20{name(first,last)}}&variables=
{"limit":100}&operationName=fetchAccounts
```

- **POST** with a JSON-encoded body: the `Content-Type` header must be set to `application/json`, and the GraphQL query must be sent in the `query` key of the JSON object.  
For example:

```
{
 "query": "query fetchAccounts ($!limit: Int) {accounts(limit: $!limit) {name(first,last)})",
 "variables": {"limit": 100},
 "operationName": "fetchAccounts"
}
```

The `variables` and `operationName` are optional. For more information, see [POST request](https://graphql.org/) in the GraphQL documentation at <https://graphql.org/>.

- **POST** with graphql content: the `Content-Type` header must be set to `application/graphql`; the entire body of the **POST** is treated as the GraphQL query

Note: For a request sent to the `/graphql` endpoint, the GraphQL proxy API always sends the request to the backend GraphQL server as a **POST** with a JSON-encoded body, so the backend GraphQL server must support this mechanism otherwise the call fails.

## GraphQL custom schema directives and scalar types

The API Connect DataPower® API Gateway extends the GraphQL specification by providing additional schema directives and custom scalar types.

### Custom schema directives

#### @cost

The `@cost` directive associates weight values with types or fields. Types are used to calculate type cost of a request to the GraphQL API. Fields are used to calculate field cost of a request to the GraphQL API. For more information, see [The Cost Directive](#).

#### @listSize

The `@listSize` directive applies slicing arguments to fields. For more information, see [The List Size directive](#).

#### @remove

The `@remove` GraphQL schema directive specifies conditions for removing types and fields from validation or introspection for each transaction based on values in the API context. For more information, see [@remove directive](#)

#### @scalarParam

The `@scalarParam` directive allows further customization of custom scalar types. For more information, see [@scalarParam directive](#).

### Custom scalar types

#### Date

The `Date` custom scalar type is used to define custom scalars that specify dates. For more information, see [Date Custom Scalar](#).

#### DateTime

The `DateTime` custom scalar type is used to define custom scalars that specify date and time. For more information, see [DateTime Custom Scalar](#).

#### JSON

The `JSON` custom scalar type is used to specify JSON payloads. For more information, see [JSON Custom Scalar](#).

#### Long

The `Long` custom scalar type enables the use of integers larger than those allowed by `Int`. For more information, see [Long Custom Scalar](#).

### Related reference

- [Using the GraphQL schema editor](#)

### @remove directive

The `@remove` GraphQL schema directive specifies conditions for removing types and fields from validation or introspection for each transaction based on values in the API context.

For example, you can prevent specific Plans from using certain types or fields. For more information on the API context, see [API Connect context variables](#).

Note: The `@remove` directive dynamically modifies the schema so that queries that do not conform to the schema, after any `@remove` directives have been evaluated, would be rejected by a `validate` policy.

Syntax

```
@remove(if: [String]) on SCALAR | OBJECT | FIELD_DEFINITION | ARGUMENT_DEFINITION | INTERFACE | UNION | ENUM | ENUM_VALUE | IN
```

Arguments

Argument name	Type	Description
<code>if</code>	List of strings	JSONata condition that specifies the criteria for removal of a type or field from validation or introspection.

Limitations and propagation rules

For limitations and propagation rules, see [Limitations and propagation rules for the GraphQL @remove directive](#).

#### Example 1

Remove the `creditCard` type if the current API plan is `bronze`.

```
type creditCard @remove(if: ["plan.name in [\\"bronze\\""]])
```

#### Example 2

Remove the `creditCard` type if the current API plan is not `gold`, and remove the field `card1` if the API plan is `bronze`.

```
type creditCard @remove(if: ["$not(plan.name = \\"gold\\")"]){
 field1: String
}
type User {
 card1: creditCard @remove(if: ["plan.name = 'bronze'", "$not(plan.name = \\"gold\\")"])
}
```

API Gateway only

## GraphQL limitations

This page details the limitations in the API Connect GraphQL implementation for the current release.

- Specialized handling of custom scalars is not supported, except for validation of the `Long` custom scalar type. Input values provided for all other custom scalar types are passed through.
- The following actions and functions cannot read or write to parsed GraphQL queries. Some of these policies might read and write a serialized form of the GraphQL query.
  - Assembly GatewayScript policy
  - Assembly Map policy
  - Assembly Redaction policy
  - Assembly Set Variable policy
  - Assembly XSLT policy
  - `setVar ()` function
- When reading parsed GraphQL queries, the Assembly Switch policy cannot use wildcards or read arguments.
- GraphQL schemas cannot be serialized.
- GraphQL messages received with `Content-Type: application/graphql` or `Content-Type: application/json` will send that same content type to the backend. This behavior cannot be overridden.
- All GraphQL schemas are checked for unbounded lists, including those that are not referenced in the assembly. This behavior can cause validation errors that prevent the GraphQL API from being published.
- The GraphQL schema editor generates warnings for fields that contain unbound lists of values if either of the following conditions is true.
  - The values in the list are of composite type. However, the warning is not valid if the composite type in the list has a defined type weight of 0.0 and all its sub-fields also have weights of 0.0 (both the fields and the types of the values they return).
  - The values in the list are of scalar type, for which a weight other than 0.0 was defined.
- Extending schemas is not supported.
- GraphQL APIs can be tested from the Developer Portal user interface. However, not all authentication methods are supported. You must use one of the following authentication methods:
  - Unsecured
  - Client ID, passed in the header
  - Client ID and secret, passed in the header

Also, `cost` metrics are not supported, and rate limits specific to GraphQL are not listed in Product plans.


## Importing an API from the CP4I asset repository

If you are using IBM® API Connect with IBM Cloud Pak for Integration (CP4I), you can import an API definition from the CP4I asset repository.

### About this task

To complete this task you must be assigned a role that has the `Api-Drafts:Edit` permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see [Creating custom roles](#).

### Procedure

1. In the navigation pane, click  Develop, select the APIs tab, then click Add API (from REST, GraphQL or SOAP). The Select API type screen is displayed.
2. Select OpenAPI 2.0 or OpenAPI 3.0 according to which version of the OpenAPI specification your API is to be based on.
3. In the Import section, select From asset repository, then click Next.
4. Click Launch the asset repository.
5. From the asset repository listing, click the name of the API that you want to import. An API explorer window opens, enabling you to examine the details of the API.
6. Click Create from asset. The validity of the API definition is checked, and a message is displayed indicating successful validation.
7. Click Next.
8. Optional: Select Activate API if you want to immediately use the API for further development and testing.  
Note:
  - When you select the Activate API option, API Connect automatically completes the following actions:

- Creates a draft Product, adds the API to the Product, and publishes the Product to the Sandbox Catalog so that the API is available to be called. The Product has the title *api\_title* auto product. This Product is not visible in the Develop view and you cannot delete it directly. However, if you delete the API the draft Product is deleted together with the API; see [Deleting an API definition](#). The Product is visible in any Catalogs to which it is published. If you want to remove the Product from a Catalog, you must do this separately; see [Removing a Product from a Catalog](#).
  - Subscribes the Sandbox test application to the Product so that you can immediately test the API in the test environment. For information on testing an API, see [Testing an API](#).
  - You cannot use the Activate API option if lifecycle approval is enabled in the Sandbox Catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to use the Activate API option they must be disabled; for information on lifecycle approval settings, see [Creating and configuring Catalogs](#).
  - To use the Activate API option, you must be assigned a role that has the **Product:Manage** and **Subscription:Manage** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see [Creating custom roles](#).
9. Click Next.
- The Summary panel displays messages as the API definition is created.
10. Select one of the following options:
- To further configure your API, click Edit API. For details, see [Editing an API definition](#).
  - If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see [Editing an API definition](#).

---

## Editing an OpenAPI 2.0 API definition

IBM® API Connect provides a form based editor that enables you to configure APIs that conform to the OpenAPI 2.0 specification.

### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

### Procedure

You can open an API for editing in either of the following ways:

- During the initial creation of an API, the API wizard guides you to enter the minimum configuration settings; on completion of the initial configuration, click Edit API.
- To open an existing API for editing, complete the following steps:
  1. In the navigation pane, click  Develop, then select the APIs tab.
  2. Click the title of the API version that you want to work with.
- [Specifying API metadata](#)  
API metadata provides summary information about the API. You enter API metadata in the Info section of the API definition.
- [Specifying the host for an API](#)  
You specify a host for an API if you want external developers on the portal to access your APIs by using a host address that is not the same as the API endpoints that are defined for the gateway services where your API will run.
- [Specifying the basepath for an API](#)  
The base path is the initial URL segment of the API, and does not include the host name or any additional segments for paths or operations. It is shared by all operations in the API.
- [Specifying the schemes for an API](#)  
Schemes define which transfer protocols you want your API to use. If your API is enforced by an API Connect gateway, only the HTTPS protocol is supported.
- [Specifying the MIME types that an API consumes](#)  
You can specify which types of media your API will accept when calls are made to it. The API Connect gateway supports XML and JSON.
- [Specifying the MIME types that an API produces](#)  
You can specify which types of media your API will return when calls are made to it. The API Connect gateway supports XML and JSON.
- [Enforcing security requirements on an API](#)  
To enforce security requirements on an API, you apply previously created security schemes that specify various aspects of API security configuration.
- [Specifying external documentation for an API](#)  
You can reference an external resource for extended documentation for an API.
- [Adding tags to an API](#)  
You can include tags in API definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.
- [Defining Paths for an API](#)  
A Path is a unit of a REST API that you can call. A Path comprises an HTTP verb and a URL path. By configuring the Path, you define how the API is exposed to your developers.
- [Defining schemas for an API](#)  
Schema components define reusable schemas that provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation.
- [Defining parameters for an API](#)  
Parameters can specify variable elements of a URL path, query parameters, headers, or a request body.
- [Defining responses for an API](#)  
Responses define the HTTP status code and data returned in a response body and headers. Responses defined for an API can be used across operations.
- [Specifying gateway and portal settings](#)  
Define general configuration settings for your API.
- [Defining target services for an API](#)  
A target service defines a web service that you want to use in your API definition.
- [Setting API properties](#)  
In addition to the pre-supplied API properties that you can use to control the behavior of API Connect policies, you can define your own API properties. The properties that you define can be referenced in your API definitions.

- [Configuring application authentication for an API](#)  
Application authentication settings allow you to protect your API with a certificate, for example, by using TLS mutual authentication. You can select whether a client certificate is sent as a TLS client certificate or in an HTTP header.
- [Configuring activity logging](#)  
You can configure your logging preferences for the API activity that is stored in Analytics, overriding the default activity logging behavior.
- [Configuring v5 compatibility options](#)  
You can configure settings to ensure compatibility with API Connect Version 5.0 compatible APIs that are converted for use with the DataPower® API Gateway.
- [Including elements in your API assembly](#)  
An assembly is formed of elements that are applied to calls to and responses from operations in your API. Elements can be either policies or logic constructs.
- [Specifying the gateway type for an API definition](#)  
An API definition is specific to one or other of the gateway types, DataPower API Gateway or DataPower Gateway (v5 compatible). A default gateway type is set when you create an API definition, but you can edit the API configuration to specify a different gateway type.
- [Converting an API definition for deployment to the DataPower API Gateway](#)  
IBM API Connect provides two gateway types, DataPower Gateway (v5 compatible) and DataPower API Gateway. If you have an API definition that was developed for the DataPower Gateway (v5 compatible) and you want to port it to the DataPower API Gateway, you must make changes to convert it before deployment.
- [Modifying a GraphQL schema](#)  
For a GraphQL API, an editor is provided for modifying the associated GraphQL schema.
- [Validating the OpenAPI YAML source](#)  
How to validate the OpenAPI YAML source when you're editing an API.

---

## Specifying API metadata



API metadata provides summary information about the API. You enter API metadata in the Info section of the API definition.

### About this task

---

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand the General section, then click Info.
3. Provide the following information:
  - Title (required): The title of the API.
  - Name: The name was generated automatically when the API was created and cannot be changed. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see [apic draft-apis](#).
  - Version: The API version is specified when the API is initially created and cannot be changed thereafter. You can, however, create a new version of an existing API; see [Creating a new version of an API definition](#).
  - Description: An optional description of the API. You can use [CommonMark syntax](#) for rich text representation.
  - Terms of Service: A URL to the Terms of Service for the API.
  - Contact Name: The identifying name of the contact person/organization.
  - Contact Url: A URL pointing to the contact information.
  - Contact Email: The email address of the contact person/organization.
  - License Name: The license name used for the API.
  - License Url: A URL to the license used for the API.
4. Click Save when done.

---

## Specifying the host for an API



You specify a host for an API if you want external developers on the portal to access your APIs by using a host address that is not the same as the API endpoints that are defined for the gateway services where your API will run.

### About this task

---

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand the General section, then click Host.
3. Enter the required host name.  
Note: To enable this capability, you must additionally configure API vanity endpoints in the Catalog Settings for the Catalog to which the API is published. For information on configuring vanity endpoints, see [Creating and configuring Catalogs](#)

4. Click Save when done.

---



## Specifying the basepath for an API

The base path is the initial URL segment of the API, and does not include the host name or any additional segments for paths or operations. It is shared by all operations in the API.

### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand the General section, then click Base Path.
3. Enter the required base path, preceded by a forward slash character (/).
4. Click Save when done.

---



## Specifying the schemes for an API

Schemes define which transfer protocols you want your API to use. If your API is enforced by an API Connect gateway, only the HTTPS protocol is supported.


### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand the General section then, if there are already one or more schemes defined, expand Schemes List.
3. To specify a new scheme, click the add icon  alongside Schemes List. To modify an existing scheme, click the scheme entry in the navigation pane.
4. Select the required scheme from the list provided, then click Create.
5. Click Save when done.

---



## Specifying the MIME types that an API consumes

You can specify which types of media your API will accept when calls are made to it. The API Connect gateway supports XML and JSON.


### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand the General section then, if there are already one or more consumes MIME types defined, expand Consumes.
3. To specify a new MIME type, click the add icon  alongside Consumes. To modify an existing MIME type, click the MIME type entry in the navigation pane.
4. Enter the required value in the Mime Type field; for example, `application/json` or `application/xml`. Then click Create.
5. Click Save when done.

---


## Specifying the MIME types that an API produces

You can specify which types of media your API will return when calls are made to it. The API Connect gateway supports XML and JSON.


## About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand the General section then, if there are already one or more produces MIME types defined, expand Produces.
3. To specify a new MIME type, click the add icon  alongside Produces. To modify an existing MIME type, click the MIME type entry in the navigation pane.
4. Enter the required value in the Mime Type field; for example `application/json` or `application/xml`. Then click Create.
5. Click Save when done.

## Enforcing security requirements on an API

To enforce security requirements on an API, you apply previously created security schemes that specify various aspects of API security configuration.

## About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).



You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

For details on how to create and configure security scheme definitions, see [Defining security schemes](#).


The following restrictions exist when you apply security schemes to an API:

- You cannot apply more than two API key security schemes to an API.
- If you apply an API key security scheme for client secret, you must also apply an API key security scheme for client ID.
- If you require the application developer to supply both client ID and client secret, you must apply two separate API key security schemes.
- You can have at most one API key scheme of type client ID, regardless of whether the client ID is sent in the request header or as a query parameter.
- You can have at most one API key scheme of type client secret, regardless of whether the client secret is sent in the request header or as a query parameter.
- You cannot apply more than one basic security scheme to an API. If you apply a basic security scheme, you cannot also apply an OAuth security scheme.
- You can apply at most one OAuth security scheme to an API.

A security requirement specifies one or more security schemes whose conditions must all be satisfied for the API to be called successfully. You can define multiple security requirements; in this case, an application can call your API if it satisfies any of the security requirements you have defined.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .



## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand General.
3. To create a new security requirement for the API, complete the following steps:
  - a. Click the add icon  alongside Security in the navigation pane.
  - b. Select the security schemes that you want to include in this security requirement. The security schemes listed are those that have been defined; see [Defining security schemes](#).

If a selected security scheme is of type OAuth2, select the required scopes; the scopes available for selection are those that were specified in the security scheme; for more information, see [Defining OAuth2 security schemes](#).

If you are applying the OAuth2 security scheme to an API that is enforced by the DataPower® API Gateway, you only need select any scopes if Advanced scope check after token generation is not enabled in the native OAuth provider associated with the security scheme. If a default scope has been set in the native OAuth provider and the API request doesn't contain any scope, the default scope is used; for more information, see [Configuring scopes for a native OAuth provider](#).

Note: The following additional requirement applies to security schemes that will be used with an OAuth third party provider. If you select an OAuth security scheme for protecting a consumer API, you must also include an API key security scheme, as the `X-IBM-Client-Id` or `client_id` must be included in the security credentials so that the correct Plan configuration settings can be enforced.

- c. Click Create. The security scheme selections are shown; you can change them again before saving.
  - d. Click Submit when done.
4. To modify an existing security requirement, complete the following steps:
    - a. Click Security in the navigation pane. All previously defined security requirements are listed; the security schemes included in each security requirement are shown.
    - b. To change the security schemes for a security requirement, click the edit icon  alongside the required security requirement, then change your security requirement selections as required.
    - c. Click Submit when done, then click Save.
    - d. To delete a security requirement, click the appropriate delete icon , click Delete to confirm, then click Save.
    - e. To disable security for the API, clear the Require one of the following Security Requirements check box, then click Save.

- [Defining security schemes](#)

A security scheme specifies all the settings for a particular aspect of API security; for example, the user registry that you use to authenticate access to the API.

- [Enabling CORS support for an API](#)

You can enable cross-origin resource sharing (CORS) support for your API. CORS allows embedded scripts in a web page to call the API across domain boundaries.

## Defining security schemes

A security scheme specifies all the settings for a particular aspect of API security; for example, the user registry that you use to authenticate access to the API.

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can create security definitions of the following types:

Type	Description
Basic authentication	Use a basic authentication security definition to specify a user registry or an authentication URL to be used to authenticate access to the API.
API key	Use an API key security definition to specify what application credentials are required to call an API.
OAuth2	Use an OAuth2 security definition to specify settings for OAuth token based authentication for your API.

- [Defining basic authentication security schemes](#)

A basic authentication security scheme is used when an application that calls the API is required to authenticate through a user registry.

- [Defining API key security schemes](#)

An API key security scheme is used to specify the credentials that an application must provide to identify itself when calling the API operations.

- [Defining OAuth2 security schemes](#)

An OAuth2 security scheme defines the settings for controlling access to the API operations through the OAuth authorization standard.

## Defining basic authentication security schemes

A basic authentication security scheme is used when an application that calls the API is required to authenticate through a user registry.

### Before you begin

For authenticating access to the operations of APIs that are enforced on the API Connect gateway, the following user registry types are supported:

- Authentication URL
- LDAP

The following user registry types are **not** supported:

- OpenID Connect (OIDC)
- Local User Registry (LUR)
- A custom user registry

Before you can create a basic authentication security definition in an API, the user registry must exist. To create a user registry, you can use either API Manager or Cloud Manager. When you create a registry in API Manager, it is visible only to your provider organization. When you create a registry in Cloud Manager, you can make it visible to multiple provider organizations.

To create a user registry with API Manager, see [Working with user registries](#).

To create a user registry with Cloud Manager, see [User registries overview](#).

### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.



When you use basic authentication, you require API users to provide a valid user name and password to access selected operations. The application developer must also provide an HTTP authorization header in requests that are sent to operations that require basic authentication.

When you use an authentication URL, the user credentials that are provided in the authorization header are validated by the endpoint specified in the URL. If the user is authenticated, IBM® API Connect expects an authentication URL to return an HTTP 200 OK response status code. All other HTTP response status codes result in an authentication failure and access is denied.

For more information about using an LDAP user registry for authentication, see [LDAP authentication](#).


For information about using an Authentication URL, see [Authentication URL user registry](#).

To make use of a basic authentication security scheme, you must reference it from elsewhere in your API definition. For more information, see [Enforcing security requirements on an API](#) and [Enforcing security requirements on an operation](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. If there are already one or more security schemes defined, expand Security Schemes.

- To create a basic authentication security scheme, click the add icon  alongside Security Schemes, then select basic for the Security Definition Type. To edit an existing basic authentication security scheme, click the security scheme name in the navigation pane.
- Enter an identifying name of your choice in the Name field.
- Select the user registry from the Authenticate using User Registry list.
 

The user registries in the list are those that have been enabled in the Sandbox Catalog; for details, see [Creating and configuring Catalogs](#).
- Optionally, provide a Description for the security scheme. You can use [CommonMark syntax](#) for rich text representation.
- If you are creating a new basic authentication security scheme, click Create.
 

The security scheme details are displayed for further editing.
- Click Save when done.

## What to do next

Apply the security scheme to an API or operation. For more information, see [Enforcing security requirements on an API](#) and [Enforcing security requirements on an operation](#).

For more information on LDAP and Authentication URL, see [LDAP authentication](#) and [Authentication URL user registry](#).

## Defining API key security schemes

An API key security scheme is used to specify the credentials that an application must provide to identify itself when calling the API operations.



### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).


You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can require that, when calling an API operation, an application must provide either a client ID, or a client ID and client secret; you create an API key security scheme to specify a credentials requirement. If you require that an application must provide both a client ID and client secret, you must create two API key security schemes, one for each type of credentials.

To make use of an API key security scheme, you must reference it from elsewhere in your API definition. For more information, see [Enforcing security requirements on an API](#) and [Enforcing security requirements on an operation](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

- Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
- If there are already one or more security schemes defined, expand Security Schemes.
- To create an API key security scheme, click the add icon  alongside Security Schemes, then select apiKey for the Security Definition Type. To edit an API key existing security scheme, click the security scheme name in the navigation pane.
- Enter an identifying name of your choice in the Security Scheme Name field.
- Specify the parameter name in the Name field.

If your API is not enforced by the API Connect gateway, enter the parameter name required by your gateway.

If your API is enforced by the IBM® API Connect gateway, the required value depends on the gateway type setting for the API, as follows:

- DataPower® API Gateway: For a key of type client secret, you must include the string "secret", ignoring case, in the value you specify in the Name field; if the value does not contain the string "secret" then the key is assumed to be of type client ID.
- DataPower Gateway (v5 compatible): the value must be as specified in the following table, depending on where the client credentials are located, and the type of credentials.

Table 1. Client ID and Client secret parameter name values for the DataPower Gateway (v5 compatible)

Location of credentials	Type of credentials	Parameter name
Header	Client ID	X-IBM-Client-Id
Header	Client secret	X-IBM-Client-Secret
Query	Client ID	client_id
Query	Client secret	client_secret

When you first create an API, a default API key security definition, of type client ID, is provided.

For information about including API key parameters in an API call, see [Calling an API](#).

Note:

- You cannot apply more than two API key security definitions to an API.
- If you apply an API key security definition for client secret, you must also apply an API key security definition for client ID.
- If you require the application developer to supply both client ID and client secret, you must apply two separate API key security definitions.
- You can have at most one API key definition of type client ID, regardless of whether the client ID is sent in the request header or as a query parameter.
- You can have at most one API key definition of type client secret, regardless of whether the client secret is sent in the request header or as a query parameter.
- The client ID and client secret headers that are specified in the request when the API is called are **not** added automatically to the message context. If you need these headers in the message context for subsequent processing, include a **set-variable** policy in your API assembly that adds the headers to the message content, taking the values specified in the request; you can do this in either of the following ways:
  - In the Assemble tab, add a Set Variable policy to your API assembly that defines the appropriate actions; for example:

Table 2.



Action	Set	Type	Value
Set	message.headers.X-IBM-Client-Id	string	\$(request.headers.X-IBM-Client-Id)
Set	message.headers.X-IBM-Client-Secret	string	\$(request.headers.X-IBM-Client-Secret)

For more information on configuring a Set Variable policy, see [Set Variable](#).

- In the Source tab, add a **set-variable** policy directly to the assembly section in your OpenAPI source; for example:

```
assembly:
 execute:
 - set-variable:
 version: 2.0.0
 title: set-variable
 actions:
 - set: message.headers.X-IBM-Client-Id
 value: $(request.headers.X-IBM-Client-Id)
 type: string
 - set: message.headers.X-IBM-Client-Secret
 value: $(request.headers.X-IBM-Client-Secret)
 type: string
```

For more information on defining a **set-variable** policy in your OpenAPI source, see [set-variable](#).

You should position the **set-variable** before any policy that needs to access the API key headers; for example, an invoke policy that calls a back-end service that is required to identify the application that made the initial request.

- Select the Key Type: `client_id` or `client_secret`.

Important: The `client_id` is not confidential and functions as a "SecurityScheme" only in the meaning of the OpenAPI specification.

If you use only the `client_id` (without the `client_secret`) as your API Key, be aware that the `client_id` is not considered a confidential value in the API Manager UI; for example, it might be displayed to other authenticated members of the provider organization. The `client_id` is treated as a public identifier for apps and cannot be kept private.

If authentication of the identity of the app calling an API is needed, then the API Key should use both a `client_id` to identify the app, plus a `client_secret` to authenticate it. The `client_secret` is designed to be known only to the app and the authorization server, and it should be protected. IBM API Connect protects the `client_secret`: it is only available when first generated, it can be stored as a one-way hash to prevent retrieval or leaks, and it can be changed and reset without changing the corresponding `client_id`. When an API Key is not marked as a `client_secret`, it will not be protected in the API Manager UI.

- Specify whether the credentials are sent in the request header, or as query parameters, by selecting one of the following Located In options:

header

The credentials are sent in the request header. This is the default setting.

query

The credentials are sent as query parameters. This method is less secure because the client secret could be exposed in a log file.

The selected option is enforced, and API calls fail if the credentials are included in the wrong location by the caller.

Note:

- You must specify the same location for the client ID and client secret, either header or query.
- If you migrated from API Connect V5, API key security definitions that contain multiple Client ID references, behave differently in Version 10 than in Version 5. For information about the differences, see [Client ID behavior in API key security definitions](#).

- Optionally, provide a Description for the security scheme. You can use [CommonMark syntax](#) for rich text representation.

- If you are creating a new API key security scheme, click Create.

The security scheme details are displayed for further editing.

- Click Save when done.

## What to do next

Apply the security scheme to an API or operation. For more information, see [Enforcing security requirements on an API](#) and [Enforcing security requirements on an operation](#).

- [Client ID behavior in API key security definitions](#)  
How the Client ID behaves in API key security definitions in IBM API Connect 10 compared to Version 5.

## Client ID behavior in API key security definitions

How the Client ID behaves in API key security definitions in IBM® API Connect 10 compared to Version 5.

API key security definitions that contain multiple Client ID references, behave differently in Version 10 than in Version 5. Refer to the following table to understand the differences in Client ID behavior for different use cases.

Note: Although a `client_id` in the request body is supported, this data is not used during security requirement processing.

Table 1. Client ID behavior in V5 and V10

Use case	Description of security requirement	V5 behavior	V10 behavior
1	Allows <code>client_id</code> in the query, and the request contains multiple <code>client_ids</code> in the query.	Uses the first client ID in the request.	401 invalid client ID or secret.
2	Allows <code>client_id</code> in the header, and the request contains multiple <code>client_ids</code> in the header.	401 invalid client ID or secret.	401 invalid client ID or secret.
3	Allows <code>client_id</code> in the header, and the request contains multiple <code>client_ids</code> in the query and the header (1 in the query, 1 in the header).	401 client ID in wrong location.	Client ID in query is ignored.
4	Allows <code>client_id</code> in the query, and the request contains multiple <code>client_ids</code> in the query and the header.	Client ID in header is ignored.	Client ID in header is ignored.
5	Allows <code>client_id</code> in the query or header, and both the query and header contain multiple <code>client_ids</code> .	Header returns 401 client ID in wrong location. Query uses the first client ID in the request.	401 invalid client ID or secret.

Use case	Description of security requirement	V5 behavior	V10 behavior
6	Allows client_id in the header, and the request contains a valid client_id in the query, and an invalid client_id in the header.	401 client ID in wrong location.	401 invalid client ID or secret.
7	Allows client_id in the query, and the request contains a valid client_id in the header, and an invalid client_id in the query.	401 invalid client ID or secret.	401 invalid client ID or secret.
8	Allows client_id in the query or header, and both the query and header contain valid client_ids.	200 OK.	403 multiple client IDs.
9	Allows client_id in the query or header, and both the query and header contain invalid client_ids.	401 client ID in wrong location.	401 invalid client ID or secret.
10	Allows client_id in the query or header, and both the query and header contain a client_id, and the one in the query is valid.	200 OK.	Uses the valid one 200 OK.
11	Allows client_id in the query or header, and both the query and header contain a client_id, and the one in the header is valid.	401 client ID in wrong location.	Uses the valid one 200 OK.
12	Allows client_id in the query or header, and both the query and client_id is missing in the request (no security requirement, but client security is defined in the assembly).	401 invalid client ID or secret.	<ul style="list-style-type: none"> <li>When Return V5 Responses compatibility option is off: 401 client ID is missing.</li> <li>When Return V5 Responses compatibility option is on: 401 invalid client ID or secret.</li> </ul> <p>For information about setting compatibility options, see <a href="#">Configuring v5 compatibility options</a>.</p>

## Defining OAuth2 security schemes

An OAuth2 security scheme defines the settings for controlling access to the API operations through the OAuth authorization standard.

### Before you begin

Before you can create an OAuth2 security scheme, you must:


1. Create an OAuth provider.
  - To use Cloud Manager, see [Configuring a native OAuth provider](#) or [Configuring a third-party OAuth provider](#).
  - To use API Manager, see [Configuring a native OAuth provider](#) or [Configuring a third-party OAuth provider](#).
2. Add the OAuth provider to a catalog. If you have not created any catalogs, use the Sandbox Catalog. See the [OAuth instructions step](#) in [Creating and configuring Catalogs](#).

### About this task


Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.


To make use of an OAuth2 security scheme, you must reference it from elsewhere in your API definition. For more information, see [Enforcing security requirements on an API](#) and [Enforcing security requirements on an operation](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. If there are already one or more security schemes defined, expand Security Schemes.
3. To create an OAuth2 security scheme, click the add icon  alongside Security Schemes, then select oauth2. To edit an existing OAuth2 security scheme, click the security scheme name in the navigation pane.
4. Enter an identifying name of your choice in the Security Scheme Name field.
5. Provide the following information:
  - Security Scheme Name: An identifying name of your choice.
  - OAuth Provider: Select the OAuth provider to be used.
  - Flow: Select one of the following options:
    - Implicit
    - Resource Owner
    - Application
    - Access Code

Note: The flow options that are available for selection depend on which options are supported by the selected OAuth provider.
  - Description: An optional description of the OAuth2 security scheme. You can use [CommonMark syntax](#) for rich text representation.
  - The URL endpoints (the applicable endpoints depend on the flow type):
    - Authorization Url
    - Token Url

Note: The URL endpoint values supplied here are maintained only for informative purposes, no validation or other action is applied to them by API Connect. The OAuth security that is applied depends on the specified OAuth provider.
  - Scopes (available if you are editing an existing OAuth2 security scheme):
    - To add a new scope, click Add. To edit an existing scope, click the options icon  alongside the required scope, then click Edit.

- Provide the following information:
  - Scope: The scope identifier.
  - Description: An optional description of the scope. You can use [CommonMark syntax](#) for rich text representation.
  - If you are creating a new scope, click Create.
- 6. If you are creating a new OAuth2 security scheme, click Create.  
The security scheme details are displayed for further editing.
- 7. Click Save when done.

## What to do next

Apply the security scheme to an API or operation. For more information, see [Enforcing security requirements on an API](#) and [Enforcing security requirements on an operation](#).

## Enabling CORS support for an API

You can enable cross-origin resource sharing (CORS) support for your API. CORS allows embedded scripts in a web page to call the API across domain boundaries.

### Before you begin



This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

### About this task

Note:

- When CORS is enabled, the API Gateway runs the `cors` preflow policy to handle all CORS requests that are made to the API.
- When CORS is enabled and a preflight request is received, only the following API actions are performed:
  - The `cors` preflow policy configures the appropriate response headers.
  - The response headers are set.
- When a preflight request is received, the `request.attributes.isCORSPreflight` flag is set to `true`.
- For all preflight requests, the `security` and `client-identification` preflow policies are always skipped, whether CORS is enabled or not enabled.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

To enable CORS support for an API, complete the following steps:

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Select the Gateway tab, expand Gateway and portal settings, then click CORS.  
The CORS page opens.
3. Select Enabled.
4. Optional: Configure a CORS policy.

Should you create a CORS policy? Review the following considerations:

- A CORS policy is an optional part of an API definition. If an API definition has no CORS policy but CORS is enabled, then CORS requests will be accepted from all Origins.  
If you want to accept CORS requests from all Origins, then enable CORS but do not add a CORS policy to the API definition.
- If you create a CORS policy, then CORS requests will only be accepted from Origins that are explicitly listed in the CORS rules contained in the CORS policy. CORS requests from any other Origins will be rejected.  
If you want to accept CORS requests from only a limited number of Origins (and might also want to configure the `Access-Control-Allow-Credentials` and `Access-Control-Expose-Headers` response headers), then you should enable CORS and create a CORS policy. Only those Origins explicitly listed in the `allow-origin` field of the CORS rules in the CORS policy will be accepted; CORS requests from any Origins not listed will be rejected.
- To configure a new CORS policy, complete the following steps:
  - Alongside Policy, click Add.
  - To include the header `Access-Control-Allow-Credentials: true` in a response, select Allow Credentials. The `Access-Control-Allow-Credentials` response header tells browsers whether to expose the response to front-end JavaScript code when the request's credentials mode (`Request.credentials`) is set to `include`.
  - To append one or more of the following values to the `Access-Control-Expose-Headers` response header, select Expose headers, and select from the following options:
    - Predefined - The predefined value of the Gateway. This option is selected by default.
    - Backend - The value of `Access-Control-Expose-Headers` from the backend response.
    - Custom - A custom string.
  - Click Create.
  - Alongside Allowed Origins click Add.
  - Enter the origin URL, then click Create. This setting indicates that the response can be shared with requesting code from the specified origin.
- To modify an existing CORS policy, click its entry on the CORS page. You can then modify individual origin URLs, add further origins, and change the Allow Credentials setting.

Example

The following example has three rules:

- Accept an Origin header of `https://example.com`, and return `Access-Control-Allow-Credentials: true` in the CORS response.
- Accept an Origin header of `http://domain.com`. No `Access-Control-Allow-Credentials` header will be returned in the response.

- Accept any of the following Origin headers:
    - `http://example2.com`
    - `http://example3.com`
    - `http://example4.com`
- No `Access-Control-Allow-Credentials` header will be returned in the response.

```
cors:
 enabled: true
 policy:
 -
 allow-origin:
 - 'https://example.com'
 allow-credentials: true
 -
 allow-origin:
 - 'http://domain.com'
 -
 allow-origin:
 - 'http://example2.com'
 - 'http://example3.com'
 - 'http://example4.com'
```

5. Click Save to save your changes.
6. Optional: To implement your own CORS solution using custom OPTIONS operations, complete the following steps:
  - a. Add the following headers to your HTTP responses:

```
Access-Control-Allow-Origin: https://<portalhostname>
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
```

Where `<portalhostname>` is your Developer Portal host name.

- b. Optional: You can proxy your API through API Connect as an enforced invoke API so that CORS is handled automatically.

Important:

- If you implement your own CORS solution, you **must** disable the CORS option described in step 3.
- CORS preflight requests are sent by using the HTTP **OPTIONS** method. Therefore, if you require these requests to be handled by the API Connect gateway then you must enable the **OPTIONS** method for all APIs that will handle preflight requests; see [Defining Paths for a REST API](#).
- **OPTIONS** requests are counted as API calls against any configured rate limit. Note that you can apply rate limits to individual operations; see [Defining rate limits for an API operation](#).

---



## Specifying external documentation for an API

You can reference an external resource for extended documentation for an API.

### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand the General section, then click External Documentation.
3. Provide the following information:
  - URL (required): The URL for the target documentation.
  - Description: An optional description of the target documentation. You can use [CommonMark syntax](#) for rich text representation.
4. Click Save when done.

---



## Adding tags to an API

You can include tags in API definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.

### About this task


Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand the General then, if there already one or more tags defined, expand Tags.

3. To add a new tag, click the add icon  alongside Tags in the navigation pane. To edit an existing tag, click the tag name in the navigation pane.
4. Provide the following information:
  - Name (required): The name of the tag.
  - Description: An optional description of the tag. You can use [CommonMark syntax](#) for rich text representation.
  - External Documentation URL (available when editing an existing tag): The URL for target documentation that describes this tag.
  - External Documentation Description (available when editing an existing tag): An optional description of the target documentation.
5. If you are creating a new tag, click Create.
6. Click Save when done.

---



## Defining Paths for an API

A Path is a unit of a REST API that you can call. A Path comprises an HTTP verb and a URL path. By configuring the Path, you define how the API is exposed to your developers.

### About this task


Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. If there are already one or more Paths defined, expand Paths.
3. To create a new Path, click the add icon  alongside Paths in the navigation pane. To edit an existing Path, click the Path in the navigation pane.
4. Provide the following information:
  - Path: A relative path to an individual endpoint.  
The path is appended to the base path to construct the full URI to access the APIs. The path must start, but not end, with the / character. A parameter at the end of the path can contain a qualifier to match one or more path levels.

If you specify just the name of the parameter, then one level of that path is matched. If you want to allow for multiple levels of the path, you can prefix the parameter with one of the following qualifiers:

- \* to indicate 0 or more occurrences
- + to indicate 1 or more occurrences

The + and \* qualifiers can only be used at the end of the path.

For example, the path:

```
/petstore/{type}/{*category}
```

matches the following paths, where only one type value is matched, but all (0 or more) categories are matched:

```
/petstore/cats
/petstore/cats/supplies
/petstore/cats/supplies/health
/petstore/cats/supplies/health/medicines
/petstore/cats/supplies/health/medicines/a/b/c
```

- \$ref (available when editing an existing Path): A reference to a Path that is defined elsewhere, either in this OpenAPI definition, or in an external file. A reference enables reuse of a set of operations already defined for a Path in this API, or in a file that is either a complete API definition or an OpenAPI fragment.

The reference uses JSON pointer notation. For example, a reference to a Path called `/mypath` in this API would have the following format:

```
#/paths/~mypath
```

The `~1` characters are used to escape the / character, which is being used literally in the path name rather than as a special character.

A reference to a path in an external file called `path_fragments.yaml` would have the following format:

```
file://path_fragments.yaml/paths/~mypath
```

5. If you are creating a new Path, click Create.  
The Path details are displayed for further editing as described in step 4.
  6. Click Save when done.
- [Defining parameters for a path](#)  
Path parameters can specify variable elements of a URL path, query parameters, headers, or a request body. The parameters apply to all the operations defined under the path. A Path parameter can be overridden at the operation level but cannot be removed there.
  - [Defining operations for a path](#)  
Operations on a Path defines the mechanisms available for interacting with the API when it is called using that Path. An operation can be of various pre-defined types; for example, a POST operation is most commonly used for creating new resources, while a GET operation is used to retrieve a resource.

---

## Defining parameters for a path



Path parameters can specify variable elements of a URL path, query parameters, headers, or a request body. The parameters apply to all the operations defined under the path. A Path parameter can be overridden at the operation level but cannot be removed there.

## About this task

---


Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand Paths.
3. Expand the required Path, then, if there already one or more parameters defined, expand Path Parameters.
4. You can either create a new parameter, or edit any existing parameter.
  - To create a parameter, click the add icon  alongside Path Parameters, then refer to [Creating a parameter](#).
  - To edit an existing parameter, click the parameter name in the navigation pane, then refer to [Editing a parameter](#).

## Defining operations for a path



Operations on a Path defines the mechanisms available for interacting with the API when it is called using that Path. An operation can be of various pre-defined types; for example, a POST operation is most commonly used for creating new resources, while a GET operation is used to retrieve a resource.

## About this task

---


Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand Paths, then expand the required path.
3. Click the add icon  alongside Operations under the Path in the navigation pane.
4. Provide the following information:
  - Verb: Select the operation type from the following options:
    - get
    - post
    - put
    - delete
    - options
    - head
    - patch
    - trace
  - Summary: A summary of what the operation does.
  - Description: A description of the operation behavior.
  - Operation: Unique string used to identify the operation. The id must be unique among all operations described in the API.
  - Deprecated: Declares this operation to be deprecated. Consumers should refrain from using the declared operation.
5. Click Create.
6. Click Save when done.

- [Adding tags to an operation](#)

You can include tags in API operation definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.

- [Specifying external documentation for an operation](#)

You can referencing an external resource for extended documentation for an API operation.

- [Specifying the MIME types that an API operation produces](#)

You can specify which types of media your API operation will return when calls are made to it. The API Connect gateway supports XML and JSON.

- [Specifying the MIME types that an API operation consumes](#)

You can specify which types of media your API operation will accept when calls are made to it. The API Connect gateway supports XML and JSON.

- [Defining parameters for an operation](#)

Responses define the HTTP status code and data returned in a response body and headers.

- [Defining responses for an operation](#)

Responses define the HTTP status code and data returned in a response body and headers. A response defined for an operation is one that can returned by executing the operation.

- [Specifying schemes for an operation](#)

The schemes settings define which transfer protocols you want your API operation to use.

- [Enforcing security requirements on an operation](#)

To enforce security requirements on an API operation, you apply previously created security schemes that specify various aspects of API security configuration.

---

## Adding tags to an operation



You can include tags in API operation definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.

### About this task

---


Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand Paths, then expand the required Path.
3. Expand Operations, then expand the required operation.
4. To add a new operation tag, click the add icon  alongside the Tags entry for the operation in the navigation pane. To edit an existing tag, expand Tags then click the tag name in the navigation pane.
5. Provide the following information:
  - Name (required): The name of the tag.
6. If you are creating a new operation tag, click Create.
7. Click Save when done.

---

## Specifying external documentation for an operation



You can referencing an external resource for extended documentation for an API operation.

### About this task

---

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand Paths, then expand the required Path.
3. Expand Operations, expand the required operation, then click External Documentation.
4. Provide the following information:
  - URL (required): The URL for the target documentation.
  - Description: An optional description of the target documentation. You can use [CommonMark syntax](#) for rich text representation.
5. Click Save when done.

---

## Specifying the MIME types that an API operation produces



You can specify which types of media your API operation will return when calls are made to it. The API Connect gateway supports XML and JSON.

### About this task

---


Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand Paths, then expand the required Path.
3. Expand Operations, then expand the required operation.
4. To specify a new MIME type, click the add icon  alongside Produces. To modify an existing MIME type, click the MIME type entry in the navigation pane.
5. Enter the required value in the Mime Type field; for example `application/json` or `application/xml`. Then click Create.
6. Click Save when done.

---

## Specifying the MIME types that an API operation consumes



You can specify which types of media your API operation will accept when calls are made to it. The API Connect gateway supports XML and JSON.

### About this task

---


Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand Paths, then expand the required Path.
3. Expand Operations, then expand the required operation.
4. To specify a new MIME type, click the add icon  alongside Consumes. To modify an existing MIME type, click the MIME type entry in the navigation pane.
5. Enter the required value in the Mime Type field; for example, `application/json` or `application/xml`. Then click Create.
6. Click Save when done.

---

## Defining parameters for an operation



Responses define the HTTP status code and data returned in a response body and headers.

### About this task

---


Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand Paths, then expand the required Path.
3. Expand Operations, then expand the required operation.
4. You can either create a new operation parameter, or edit any existing parameter.
  - To create an operation parameter, click the add icon  alongside Path Parameters, then refer to [Creating a parameter](#).
  - To edit an existing operation parameter, click the parameter name in the navigation pane, then refer to [Editing a parameter](#).

---

## Defining responses for an operation



Responses define the HTTP status code and data returned in a response body and headers. A response defined for an operation is one that can returned by executing the operation.

### About this task

---


Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand Paths, then expand the required Path.
3. Expand Operations, expand the required operation, then, if there are any existing responses, expand Responses.
4. You can either create a new response, or edit any existing response.
  - To create a response, click the add icon  alongside Responses, then refer to [Creating a response](#).
  - To edit an existing response, click the response name in the navigation pane, then refer to [Editing a response](#).

---

## Specifying schemes for an operation



The schemes settings define which transfer protocols you want your API operation to use.

## About this task

---


Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Expand Paths, then expand the required Path.
3. Expand Operations, expand the required operation, then, if there are any existing schemes, expand Schemes List.
4. You can either add a new scheme, or edit any existing scheme.
  - To add a scheme, click the add icon  alongside Schemes List, select the required transfer protocol, then click Create.  
Note: If your API is enforced by an API Connect gateway, only the HTTPS protocol is supported.
  - To edit an existing scheme, click the scheme in the navigation pane, then change the transfer protocol as required.
5. Click Save when done.

## Enforcing security requirements on an operation

---

To enforce security requirements on an API operation, you apply previously created security schemes that specify various aspects of API security configuration.

### About this task

---



Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

For details on how to create and configure security schemes, see [Defining security schemes](#).




A security requirement specifies one or more security scheme definitions whose conditions must all be satisfied for the API operation to be called successfully. You can define multiple security requirements; in this case, an application can call your API operation if it satisfies any of the security requirements you have defined.

Any security requirements that you define for an operation completely override any security requirements defined on the parent API. If you do not define any security requirements for an operation, or you delete all security requirements from an operation, the operation inherits the security requirements defined on the parent API. For more information, see [Enforcing security requirements on an API](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
  2. Expand Paths, then expand the required Path.
  3. Expand Operations, then expand the required operation.
  4. To create a new security requirement for the operation, complete the following steps:
    - a. Click the add icon  alongside the Security entry for the operation in the navigation pane.
    - b. Select the security schemes that you want to include in this security requirement. The security schemes listed are those that have previously been defined; see [Defining security schemes](#).  
If a selected security scheme is of type OAuth2, select the required scopes; the scopes available for selection are those that were specified in the security scheme; for more information, see [Defining OAuth2 security schemes](#).  
  
If you are applying the OAuth2 security scheme to an API that is enforced by the DataPower® API Gateway, you only need select any scopes if Advanced scope check after token generation is not enabled in the native OAuth provider associated with the security scheme. If a default scope has been set in the native OAuth provider and the API request doesn't contain any scope, the default scope is used; for more information, see [Configuring scopes for a native OAuth provider](#).
  - Note: The following additional requirement applies to security schemes that will be used with an OAuth third party provider. If you select an OAuth security scheme for protecting a consumer API, you must also include an API key security scheme, as the `X-IBM-Client-Id` or `client_id` must be included in the security credentials so that the correct Plan configuration settings can be enforced.
  - c. Click Create. The security scheme selections are shown; you can change them again before saving.
  - d. Click Save when done.
  5. To modify an existing security requirement, complete the following steps:
    - a. Click the Security entry for the operation in the navigation pane. All previously defined security requirements are listed; the security schemes included in each security requirement are shown.
    - b. To change the security schemes for a security requirement, click the edit icon  alongside the required security requirement, then change your security requirement selections as required.
    - c. Click Submit when done, then click Save.
    - d. To delete a security requirement, click the appropriate delete icon , click Delete to confirm, then click Save.
    - e. To disable security for the operation, clear the Require one of the following Security Requirements check box, then click Save.
- Note: These settings completely override any security requirements defined on the parent API; see [Enforcing security requirements on an API](#).

---

## Defining schemas for an API



Schema components define reusable schemas that provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation.

### About this task

---


Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
  2. If there are already one or more schemas defined, expand Definitions.
  3. You can create a new schema, or edit any existing schema.
    - To create a schema, click the add icon  alongside Definitions, then refer to [Creating a schema](#).
    - To edit an existing schema, click the schema component name in the navigation pane, then refer to [Editing a schema](#).
- **Creating a schema**  
Schemas provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation. You can create schemas in various places in your API definition.
  - **Editing a schema**  
Schemas provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation. You can edit schemas that have been previously created in various places in your API definition.

---

## Creating a schema

Schemas provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation. You can create schemas in various places in your API definition.

### Before you begin

---

Note: Schemas are compiled before they are used for validation. Because the compilation process is longer than the validation process, the compiled schema artifacts are stored in a cache. The limited capacity of the cache can cause older entries to be evicted from the cache when newer entries are added. Schemas whose artifacts have been evicted from the cache must be recompiled, which can cause significant delays in validation.

Launch the schema creation window. For details of the areas in your API definition from where you can create a schema, see the following topics:



- [Defining schemas for an API](#)
- [Editing a response](#)

### About this task

---

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Provide the following information:
  - Schema Name: This name defines a key that enables this schema to be referenced from elsewhere in the API definition; the reference has the following format:  
**`#/definitions/Name`**
  - Title: The schema title.
  - Type: The schema data type; select one of the following:
    - array
    - boolean
    - integer
    - number
    - object
    - string
  - Description: An optional description of the schema. You can use [CommonMark syntax](#) for rich text representation.
  - Select the following options as required:
    - Read Only: A property associated with this schema can be sent in a response but should not be sent in a request.
    - Write Only: A property associated with this schema can be sent in a request but should not be sent in a response.
    - Deprecated: The schema is deprecated and should be transitioned out of usage.

2. Click Create.

The schema details are displayed for further editing; see [Editing a schema](#).

---

## Editing a schema

Schemas provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation. You can edit schemas that have been previously created in various places in your API definition.

---

### Before you begin

Note: Schemas are compiled before they are used for validation. Because the compilation process is longer than the validation process, the compiled schema artifacts are stored in a cache. The limited capacity of the cache can cause older entries to be evicted from the cache when newer entries are added. Schemas whose artifacts have been evicted from the cache must be recompiled, which can cause significant delays in validation.

Open the details form for a schema. For details of the areas in your API definition from where you can edit a schema, see the following topics:



- [Defining schemas for an API](#)
- [Editing a response](#)

---

### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

### Procedure


You can configure the following schema settings:

- Schema Name: This name defines a key that enables this schema to be referenced from elsewhere in the API definition; the reference has the following format:

**`#/definitions/Name`**

To change the name, click Update, change the value in the field, then click Save.

- Title: The schema title.
- Type: The schema data type; select one of the following:
  - - array
    - boolean
    - integer
    - number
    - object
    - string

Note: If you change the type of an existing schema, any existing settings that are not relevant to the new type are still retained in the OpenAPI source even though they are no longer displayed in the user interface. You should therefore avoid porting a schema from one type to another but instead either create a new schema or modify the OpenAPI source directly; to open the source editor, click the Source icon .

- Additional schema settings dependent on the selected Type, as follows:
  - array:
    - Max Items: The maximum allowed number of items in an array instance.
    - Min Items: The minimum allowed number of items in an array instance.
    - Unique Items: If selected, all items in an array instance must be unique.
  - boolean: no type specific settings.
  - integer:
    - Format: An optional data type modifier. For more information, see [Data Types](#) in the [OpenAPI Specification](#).
    - Multiple Of: Division of an integer instance by this value must result in an integer.
    - Maximum: An integer instance must be less than or equal to this value.
    - Exclusive Maximum: An integer instance must be strictly less than this value.
    - Minimum: An integer instance must be greater than or equal to this value.
    - Exclusive Minimum: An integer instance must be strictly greater than this value.
  - number:
    - Format: An optional data type modifier. For more information, see [Data Types](#) in the [OpenAPI Specification](#).
    - Multiple Of: Division of a number instance by this value must result in an integer.
    - Maximum: A number instance must be less than or equal to this value.
    - Exclusive Maximum: A number instance must be strictly less than this value.
    - Minimum: A number instance must be greater than or equal to this value.
    - Exclusive Minimum: A number instance must be strictly greater than this value.
  - object
    - Properties: To add properties to the schema, click Add alongside the Properties section heading. You configure a property by defining its schema with the schema editor.  
To create the property, refer to [Creating a schema](#).
  
    - To further configure the property, follow the steps on this page.

By defining properties, you can create more complex data structures. For example, you could define an "Address" schema that has the following properties:

PROPERTY NAME	PROPERTY TYPE
street1	string
street2	string
city	string
state	string
zip_code	string

and then define a "BankBranch" schema that has the following properties, one of which is a reference to the Address schema:

PROPERTY NAME	PROPERTY TYPE
address	address
type	string
id	string

- Max Properties: The number of properties in an object instance must be less than or equal to this value.
- Min Properties: The number of properties in an object instance must be greater than or equal to this value.
- Required Properties: An array of property names; every property name in the array must be the name of a property in an object instance. To add a required property name, click Add alongside the Required Properties section heading, enter the name, then click Create.
- string:
  - Format: An optional data type modifier. For more information, see [Data Types](#) in the [OpenAPI Specification](#).
  - Pattern: A regular expression; a string instance must successfully match the expression.
  - Max Length: The maximum allowed number of characters in a string instance.
  - Min Length: The minimum allowed number of characters in a string instance.
  - Enum: An array of string values; a string instance must match one of the array values. To add an enum value, click Add, enter the value, then click Create.
- Description: An optional description of the schema. You can use [CommonMark syntax](#) for rich text representation.
- General settings; select the required options:
  - Read Only: A property associated with this schema can be returned in a response but should not be included in a request. For example, properties whose values are system generated identifiers will be defined as Read Only because application clients wouldn't be able to set these.
  - Write Only: A property associated with this schema can be included in a request but should not be returned in a response. For example, password properties are usually defined as Write Only because application clients would supply these when creating a user object but shouldn't be able see them when retrieving the object.
  - Deprecated: The schema is deprecated and should be transitioned out of usage.
- Example Value: An example value. Whatever you enter here is displayed as-is in the Developer Portal.
- External Documentation: An external resource for extended documentation; provide the following information:
  - URL (required): The URL for the target documentation.
  - Description: An optional description of the target documentation. You can use [CommonMark syntax](#) for rich text representation.
- Xml: Metadata that allows for more fine-tuned XML model definitions; provide the following information:
  - Name: Replaces the name of the element or attribute used for the described schema property.
  - Namespace: The URL of the namespace definition.
  - Prefix: The prefix to be used for the Name.
  - Attribute: Specifies whether the property definition translates to an attribute instead of an element.
  - Wrapped: For an array definition, Specifies whether the array is wrapped (for example, `<books><book/></books>`) or unwrapped (`<book/></book/>`).
- Click Save when done.

## What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.



## Defining parameters for an API

Parameters can specify variable elements of a URL path, query parameters, headers, or a request body.


### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. If there already one or more parameters defined, expand Parameter Definitions.
3. You can either create a new parameter, or edit any existing parameter.
  - To create a parameter, click the add icon  alongside Parameters, then refer to [Creating a parameter](#).
  - To edit an existing parameter, click the parameter name in the navigation pane, then refer to [Editing a parameter](#).
- [Creating a parameter](#)  
Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can create parameters for Paths and Path operations in your API definition.
- [Editing a parameter](#)  
Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can edit parameters that have been previously created for the API, or for Paths and Path operations in your API definition.

---

## Creating a parameter

Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can create parameters for Paths and Path operations in your API definition.

---

### Before you begin

Launch the parameter creation window. For details of the areas in your API definition from where you can create a parameter, see the following topics:

- [Defining parameters for a path](#)
- [Defining parameters for an operation](#)
- [Defining parameters for an API](#)

---



### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A parameter is similar to a header, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **header**.
- A parameter can have various format styles, the style of a header is always **simple**.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

### Procedure

1. Provide the following information:

- Parameter Name: The name of the parameter.
- Located In: The location of the parameter. Select one of the following options:
  - query: Parameters that are appended to the URL. For example, `url_path?myparam=myvalue`.
  - header: Custom headers that are expected as part of the request.
  - path: The parameter value is part of the operation's URL, enclosed in {}.
  - body: Parameters that describe the body of POST, PUT and PATCH requests.
- Description: A description of the parameter.
- Type: The data type of the parameter.
- Format: Select the data format; the available options depend on the selected Type.
- Select the following option as required:
  - Required: Determines whether this parameter is mandatory. If the Located In property is set to path, this option must be selected.
  - Allow Empty Value: Allows sending a parameter with an empty value. This is valid only if the Located In property is set to query.

2. Click Create.

The parameter details are displayed for further editing; see [Editing a parameter](#).

---

## Editing a parameter

Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can edit parameters that have been previously created for the API, or for Paths and Path operations in your API definition.

---

### Before you begin

Open the details form for a parameter. For details of the areas in your API definition from where you can edit a parameter, see the following topics:

- [Defining parameters for a path](#)
- [Defining parameters for an operation](#)
- [Defining parameters for an API](#)

---



### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A parameter is similar to a header, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **header**.
- A parameter can have various format styles, the style of a header is always **simple**.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Provide the following information:
  - Description: A description of the parameter. You can use [CommonMark syntax](#) for rich text representation.
  - Type: The data type of the parameter.
  - Format: Select the data format; the available options depend on the selected Type.
  - Select the following options as required:
    - Required: Determines whether this parameter is mandatory. If the Located In property is set to path, this option must be selected.
    - Allow Empty Value: Allows sending a parameter with an empty value. This is valid only if the Located In property is set to query.
2. Click Save when done.

## What to do next

---

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

## Defining responses for an API

---



Responses define the HTTP status code and data returned in a response body and headers. Responses defined for an API can be used across operations.

### About this task

---


Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
  2. If there already one or more responses defined, expand Response Definitions.
  3. You can either create a new response, or edit any existing response.
    - To create a response, click the add icon  alongside Response Definitions, then refer to [Creating a response](#).
    - To edit an existing response, click the response name in the navigation pane, then refer to [Editing a response](#).
- [Creating a response](#)  
Responses define the HTTP status code and data returned in a response body and headers. You can create responses for the Path operations in your API definition.
  - [Editing a response](#)  
Responses define the HTTP status code and data returned in a response body and headers. You can edit responses that have been previously created for Path operations in your API definition.

## Creating a response

---

Responses define the HTTP status code and data returned in a response body and headers. You can create responses for the Path operations in your API definition.

### Before you begin

---

Launch the response creation window. For details of the areas in your API definition from where you can create a response, see the following topics:


- [Defining responses for an API](#)
- [Defining responses for an operation](#)

### About this task

---

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Provide the following information:
  - Name: The HTTP status code.
  - Description (required): A description of the response.
2. Click Create.  
The response details are displayed for further editing; see [Editing a response](#).

---

## Editing a response

Responses define the HTTP status code and data returned in a response body and headers. You can edit responses that have been previously created for Path operations in your API definition.

---

### Before you begin

Open the details form for a response. For details of the areas in your API definition from where you can edit a response, see the following topics:



- [Defining responses for an API](#)
- [Defining responses for an operation](#)

---

### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

### Procedure

1. Provide the following information:
  - Name: The HTTP status code.
  - Description (required): A description of the response.
  - Schema: The schema defines the response structure.  
To define a schema for the response, click Create, then refer to [Creating a schema](#).

If the response already has a schema defined, click View to edit the schema.

For full details on editing a schema, see [Editing a schema](#).
2. Click Save when done.

---

### What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

---

## Specifying gateway and portal settings



Define general configuration settings for your API.

---

### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Select the Gateway tab, then click Gateway and portal settings.
3. Provide the following information:
  - Target Services: Define web services that you want to use in your API definition. For details, see [Defining target services for an API](#).
  - Enforced: Select this option to enforce the API by using the DataPower® API Gateway. Clear this option if you are managing the API on a gateway other than the DataPower API Gateway. Although not published to a gateway by API Connect, an unenforced API is still available in the Developer Portal for subscription by application developers.
  - Phase: The phase of the lifecycle that your API is in; select one of the following options:
    - Realized (default) - The API is in the implementation phase.
    - Identified - The API is in the early conceptual phase and is neither fully designed nor implemented.
    - Specified - The API has been fully designed and passed an internal milestone but has not yet been implemented.
  - Testable: Select this option to allow the API's operations to be tested using the test tool in the Developer Portal.  
Note: For the test tool to work, an API must be included in a Plan in a Product that is staged in a Catalog.
  - CORS: Enable and configure cross-origin resource sharing (CORS) support for your API. For details, see [Enabling CORS support for an API](#).
  - Properties: Add properties that can be referenced in your API definition. For details, see [Setting API properties](#).
  - Catalog Properties: Define property values that are specific to a particular Catalog. For details, see [Defining Catalog specific property values](#).
  - Gateway: For details, see [Specifying the gateway type for an API definition](#).
  - Application Authentication: Enable application authentication to protect your API with a certificate. For details, see [Configuring application authentication for an API](#).

- Application Authentication Source: If application authentication is enabled, specify how the client certificate is sent to the gateway. For details, see [Configuring application authentication for an API](#).
- Activity Log: Configure your logging preferences for the API activity that is stored in Analytics. For details, see [Configuring activity logging](#).
- Compatibility: For details, see [Configuring v5 compatibility options](#).
- HTML Page: For a GraphQL API, the HTML page with which the API responds to an initial GraphQL editor request. Currently, the only option is `store:///graphql.html`. For more information, see [Enabling the GraphQL editor for a GraphQL API](#).
- GraphQL Schema: For a GraphQL API, the name of the schema definition that references the imported GraphQL schema. For information on uploading a schema during GraphQL API creation, see [Creating a GraphQL proxy API](#). For information on replacing the schema for an existing GraphQL API, see [Using the GraphQL schema editor](#).
- Extensions: Extensions added to the API to extend the OpenAPI specification. For details, see [Adding an OpenAPI extension to an API](#).

---



## Defining target services for an API

A target service defines a web service that you want to use in your API definition.


### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Select the Gateway tab, expand Gateway and portal settings then, if there are already one or more target services defined, expand Target Services.
3. To define a new API target service, complete the following steps:
  - a. Click the add icon  alongside Target Services in the navigation pane.
  - b. Upload the service information from a stand-alone .wsdl file, or a .zip file that contains a WSDL file and its dependent documents, by either dragging and dropping your file, or browsing and selecting the file that you want to use.  
If you upload a .zip file, you can include in the .zip file an options file to specify additional directives. For details, see [Using an options file when importing a WSDL service](#).

The target services that you add are now available in the palette on the Policies view for you to add to your API assembly flow; for more information, see [Including elements in your API assembly](#).

4. Select the service that you want to add.
5. Click Save to save your changes.
6. To view the source for an existing target service, click the target service name in the navigation pane.

### Results

The target services that you add are now available in the palette on the Policies view for you to add to your API assembly flow; for more information, see [Including elements in your API assembly](#).

---

## Setting API properties

In addition to the pre-supplied API properties that you can use to control the behavior of API Connect policies, you can define your own API properties. The properties that you define can be referenced in your API definitions.

### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).



You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

API properties include property name, value, and, optionally a specific Catalog to which a property value applies. For a list of pre-supplied API properties relating to various policies, see [API properties](#).

Note: Once defined, an API property is read only.

For information on how to reference a property in an API definition, see [Variable references in API Connect](#).


It is also possible to define properties that are specific to a Catalog and can be referenced by any of the APIs in that Catalog; for more information, see [Creating and configuring Catalogs](#). Note that if you define a Catalog property of the same name as an API property, the API property takes precedence over the Catalog property.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Tip: If you add or change an API property on an API that is already staged or published, you must re-stage or re-publish the Product that contains the updated API for the change to take effect.

### Procedure



1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
  2. Select the Gateway tab, expand Gateway and portal settings then, if there are already one or more API properties defined, expand Properties.
  3. Configure an API property.
    - To define a new API property, complete the following steps:
      - Click the add icon  alongside Properties in the navigation pane.
      - Provide the following information:
        - Property Name: Enter a name for the property; this name is used to reference the property. The following character set is supported for the name of an API property: `[A-Za-z0-9_-]+`. Spaces are allowed.
        - Property Value: A default value for the property. Leave blank if the property is to have a null value by default.
        - Description: An optional description of the property. You can use [CommonMark syntax](#) for rich text representation.
        - Select Property Value is Base64 Encoded if you want to hide the property values, or protect user passwords from casual observance.
 

Note: If you encode a property value, it is saved in Base64 encoded form; it is **not** encrypted. If you subsequently clear the Encoded check box, the original property value is restored in its unencoded form.
      - Click Create.
    - To modify an existing API property, click the property name in the navigation pane. You can then change any of the configuration settings.
  4. Click Save to save your changes.
- [Defining Catalog specific property values](#)  
For any API property, you can define a value that is specific to a particular Catalog. For any API published to that Catalog, the API property assumes the Catalog specific value, overriding any default property value.

---

## Defining Catalog specific property values

For any API property, you can define a value that is specific to a particular Catalog. For any API published to that Catalog, the API property assumes the Catalog specific value, overriding any default property value.

---

### Before you begin



You must have previously created the API property that you now want to define a Catalog specific value for; see [Setting API properties](#).

---

### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).


You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Tip: If you add or change an API property on an API that is already staged or published, you must re-stage or re-publish the Product that contains the updated API for the change to take effect.

---

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Select the Gateway tab, expand Gateway and portal settings then, if there are already one or more Catalog specific property values defined, expand Catalog Properties.
3. Configure a Catalog specific property value.
  - To define a new Catalog specific property value, complete the following steps:
    - Click the add icon  alongside Catalog Properties in the navigation pane.
    - Type the Catalog Name exactly as it displays in the **name** field for the selected Catalog.
 

Tip: For the best result, manually type the catalog's name instead of selecting its title. Using the catalog name is safer than using its title, because the title might not be unique within the provider organization and might be modified later (which breaks the association between the property and the catalog). The catalog's name is always unique within the provider organization and cannot be changed.
    - Click Create. The Catalog properties details for the specified catalog are displayed for further editing.
    - Alongside Property Overrides, click Add.
    - Provide the following information:
      - Property Name to Override: Enter the name of the API property that you want to define a Catalog specific value for. The name **must** match the name of a previously defined API property; see [Setting API properties](#).
      - Property Value: The Catalog specific property value. Leave blank if the property is to have a null value by default.
    - Click Create.
  - To modify an existing Catalog specific property value, click the Catalog name in the navigation pane. You can then change the Catalog name, and add, modify, or delete property override settings.
4. Click Save to save your changes.

---

## Configuring application authentication for an API



Application authentication settings allow you to protect your API with a certificate, for example, by using TLS mutual authentication. You can select whether a client certificate is sent as a TLS client certificate or in an HTTP header.

---

### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
  2. Enable application authentication, as follows:
    - a. Select the Gateway tab, expand Gateway and portal settings, then click Application Authentication.
    - b. Select Certificate.
  3. Specify how the client certificate is sent to the gateway, as follows:
    - a. In the navigation pane, click Application Authentication Source. Any existing application authentication source definitions are listed.
    - b. Alongside Application Authentication Source, click Add.
    - c. Select the required option, header or tls-cert. You can separately add one or both of the options. When the API is called, an X509 client certificate must be supplied, either in the specified HTTP header, or as a TLS client certificate from TLS mutual authentication. For any Developer Portal application that calls the API, the certificate must be entered in the Developer Portal user interface; for details, see [Registering an application](#).  
  
The Gateway service to which the API is published can be configured to use TLS mutual authentication to secure API calls made to that Gateway service; for details, see [Configuring the initial Gateway service](#). If you select this option, continue to sub-step d.  
  
If you are using a load balancer, you must configure the load balancer to use the specified HTTP header to relay the appropriate client certificate to the Gateway service after the load balancer terminates the TLS communication.
    - d. If you select the tls-cert option, enable mTLS at the Gateway level. This can be done by enabling "Mutual authentication: Required" in the TLS Server Profile configured in the Gateway Service in the Cloud Manager Topology (requires cloud administrator permissions). For information, see [Registering a gateway service](#).  
  
You can create a Truststore (with the root or intermediate CA certificate) for use with the TLS Server Profile.  
  
The following example cURL command uses a client-cert during SSL:

```
curl -X GET 'https://GW-Server:9443/test/sandbox/mtls-testing/callmtls-call' -H 'X-IBM-Client-Id: 1872xxxxxxxxxxxx' --cert my-leaf-cert.pem --key my-leaf-cert.key -k -v
```
- Note: This feature is intended for the scenario where different applications will be registered with different end certificates in the Developer Portal, which will have the same intermediate or root CA issuer. The certificate will pass Gateway-level mTLS if it matches the issuer available in the Truststore, but will only allow access to the API that matches the correct end certificate.
4. Click Create.
  5. Click Save to save your changes.

## Configuring activity logging

You can configure your logging preferences for the API activity that is stored in Analytics, overriding the default activity logging behavior.

## About this task



Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

An API event record exists for each API execution event in the Gateway server. By default, the content type for a successful API execution is activity, and payload for an API call the results in an error code. When you compose your API definition, you can change the type of content to log in these API event records. During API execution, the activity data is stored in the `log` context variable, which populates the API event record on completion of the API execution; for more information, see [API activity logging context variables](#).

Note:

Activity logging that calls for logging of Analytics data upon success doesn't apply for the OAuth provider. The OAuth provider logs Analytics data for failure cases, but doesn't log successful cases.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Select the Gateway tab, expand Gateway and portal settings, then click Activity Log.
3. Select Enabled, then select options for the Success Content and Error Content fields as follows:

Field label	Description
-------------	-------------

Field label	Description
Success Content	<p>Defines the type of content to be logged when the operation is successful.</p> <ul style="list-style-type: none"> <li>• none: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal.</li> <li>• activity: Logs invocation only; only the resource URI is recorded.</li> <li>• header: Logs activity and header; the HTTP headers, in addition to the resource URI are recorded.</li> <li>• payload: Logs activity, header, and payload; all information, including the payload that is received in a request or returned in a response, is recorded.</li> </ul> <p>The default value is activity.</p>
Error Content	<p>Indicates what content to log when an error occurs.</p> <ul style="list-style-type: none"> <li>• none: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal.</li> <li>• activity: Logs invocation only; only the resource URI is recorded.</li> <li>• header: Logs activity and header; the HTTP headers, in addition to the resource URI are recorded.</li> <li>• payload: Logs activity, header, and payload; all information, including the payload that is received in a request or returned in a response, is recorded.</li> </ul> <p>The default value is payload.</p>
Preserved request header	Click Add to specify customized request headers to preserve for logging to analytics.
Preserved response header	Click Add to specify customized response headers to preserve for logging to analytics.

4. Click Save to save your changes.



## Configuring v5 compatibility options

You can configure settings to ensure compatibility with API Connect Version 5.0 compatible APIs that are converted for use with the DataPower® API Gateway.

### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Select the Gateway tab, expand Gateway and portal settings, then click Compatibility.
3. Select from the following options as required:
  - Return V5 Responses: Return v5-compatible responses, such as OAuth and client security error responses.
  - Copy Id Headers to Message: Copy security headers to the message context for retrieval by the invoke back-end service.
  - Enforce Required Params: Check the request for required parameters during API routing. When enabled, a request that does not provide a required parameter is rejected.
  - Allow Chunked Uploads: Allow the assembly `invoke` policy to send documents to the server with `Transfer-Encoding: chunked`. This setting applies only to the `invoke` 1.5.0 wrapper policy deployed from API Connect using the migration utility. It does not apply to the native API Gateway assembly `invoke` policy.
4. Use the YAML source to set the following options as required:

Option	Valid values	Description
<code>allow-trailing-slash</code>	<code>true / false</code>	Specify whether to allow the request URL to end with a / character. The default value is <code>false</code> .
<code>get-raw-body-value</code>	<code>true / false</code>	Specify whether the GatewayScript <code>apim.getvariable()</code> API returns the raw body instead of parsing. This setting applies only when the context is other than <code>message</code> . The default value is <code>false</code> .
<code>request-headers</code>	<code>true / false</code>	Specify whether to populate v5-compatible headers such as <code>X-Client-IP</code> and <code>X-Global-Transaction-ID</code> in the <code>request.headers</code> context variable. The default value is <code>false</code> .

5. Click Save to save your changes.

## Including elements in your API assembly

An assembly is formed of elements that are applied to calls to and responses from operations in your API. Elements can be either policies or logic constructs.

### About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

In the Policies view of IBM® API Connect, you can add and configure elements in your assembly. You can also directly add elements to the OpenAPI definition of your API.

- **The assembly editor**  
The API Designer of IBM API Connect features a graphical editor that you can use to create assemblies. With assemblies, you can readily tailor your APIs to include elements such as activity logging and redaction of specific fields.

- [Adding elements to your assembly](#)  
Create your API assembly by using the Assemble view in the API Designer.
- [Handling errors in the assembly](#)  
Use the catch section of the assembly to describe the handling of errors thrown during the assembly execution.

---

## The assembly editor


The API Designer of IBM® API Connect features a graphical editor that you can use to create assemblies. With assemblies, you can readily tailor your APIs to include elements such as activity logging and redaction of specific fields.

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can access the Assemble view by clicking the Gateway tab in the API editor, and then click Policies. This tab includes a palette, which lists the available elements, a property sheet, which is used to configure an element, and a canvas, which is used to arrange and visualize the assembly's elements.

---

## The palette

The palette is a list of different elements that you can include in your assembly. The palette can be hidden by clicking the Show/hide policy palette icon .

---


## The canvas



You can use the canvas to create a graphical representation of the assembly flow. You can drag various elements from the palette to the appropriate location on the canvas. When you drag an element, valid positions are shown by dashed boxes.

Note: You must click Save in the Assemble view to save your assembly updates, and to make them appear in the Source view.

API calls are made at the initial, unfilled, circle, and returned at the final, filled, circle. You can insert elements between the two circles to modify the data received from the call, or returned by the response. To add an element, select it in the palette and drag it across to one of the dashed boxes that appear when you move the element over the canvas.

You can use the Show catches toggle to show and hide error catches in the palette. A catch is a section of the assembly that is applied when an API call results in the corresponding HTTP status code being returned. Click the catch section to open the property sheet for all your catches.


You can zoom the view of your canvas in and out by clicking the + and - icons. To fit the canvas to the screen size, click the Fit to screen icon .

You can filter the canvas to show only the parts of it that will apply to a specific operation by clicking the Filter by operation icon  and then selecting the operation from the drop-down list. Click the Clear operation filter icon  to remove the filter.

---

## The property sheet

When you select an element that is in the assembly by clicking it, details about the element are displayed in the property sheet. In this pane, you can configure the element's properties. The options available to you in the property sheet are specific to the type of element you are working with. For some elements, you can add and remove properties by clicking Object Properties and selecting the property from the drop-down menu.

You can release the property sheet by clicking the Close icon .

---

## The behavior of an assembly

The assembly runs policies in order and acts on different contexts of the API call. When an API call is made, security and rate limits are enforced before the assembly is executed. During the assembly, the flow can branch or be thrown and caught, according to the policies contained in it. The message context can be thought to flow through the assembly, being used and altered by various policies. In addition to the message, other contexts can be accessed and created.

### Security and rate limiting

Before the assembly is executed, security and then rate limits are enforced.

First, security definitions and CORS access control are used to authenticate an API call. Any API Key security definitions are used to identify applications that have subscriptions to a Product containing the API. If a security definition does not allow access, the API call is rejected.

If an application is identified by its client ID or client secret, a rate limit can be enforced based on the Plan or operation called.

### The assembly execution sequence

The assembly is executed in order from the initial, filled, circle to the final, unfilled, circle. However, there is room for branching, when if and operation-switch logic constructs are used, or for the remaining assembly to be ignored when a throw policy is executed.

The message is the context that is acted upon by any policy that isn't otherwise configured. At the beginning of the API call, the message is empty and, at the end of the API call, the message is used as the response.

The request context contains the information that is sent by the API caller and varies with the type of operation called and the configuration of that operation. For example, a GET operation can never have a populated `request.body`, and you can configure an operation to have `request.parameters` (query parameters).

The first policy in an assembly acts on the request and produces the first instance of the message. If there are no policies, the request is returned to the caller.

### Managing contexts

Because the message can be overwritten, it can be useful to create and reference new contexts where possible so that they are saved and reusable during the API call.

- Use the map policy to overwrite the message context when you need to execute a policy that only acts on the message.
- Use the request context when you want to use the original request made to the API.
- Use the map, invoke, and proxy policies to create new contexts when you want to save your message.

Note: When you create a new context, unless you are also mapping to the message, the message is overwritten with an empty object.

For example, an invoke policy is the first policy in the assembly and its response overwrites the request as the message. The message is then acted upon by a validate policy, and a map policy then saves the message as a new context, ready for a second invoke policy to overwrite the message without losing the first invoke policy's output.

You can also access contexts outside of the message or your custom contexts, but these cannot be written to. For a list of contexts, see [API Connect context variables](#).

#### Branches and catches

Using logic constructs, such as operation-switch or if, you can execute different sections of the assembly when certain conditions are fulfilled. When the assembly branches, the subsection of the assembly contained by the construct is executed in the same manner as a complete assembly. However, contexts are shared with the complete assembly.

When a catch is triggered, either by an error occurring during the execution of a policy, or because a throw policy is encountered, the rest of the assembly flow is ignored. All contexts are shared by the catch being executed and when the end of the catch is reached, the API call is completed. There is no way to return from a catch to the rest of the assembly.

## Related concepts

---

- [Variable references in API Connect](#)

## Related tasks

---

- [Adding elements to your assembly](#)

## Related reference

---

- [API Connect context variables](#)

# Adding elements to your assembly

Create your API assembly by using the Assemble view in the API Designer.



## About this task

---

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can use the assembly tool in the UI to create assemblies that are used to manipulate requests made to or responses made by any of your API's operations.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .


If you want to use the Source view to edit the source of your API definition, see [execute OpenAPI extension](#) for more information about how to configure the OpenAPI YAML.

For more information about the use of the assembly tool, see [The assembly editor](#).

## Procedure

---


To add elements to your assembly using the assembly tool, complete the following steps:

1. In the navigation pane, click  Develop, then select the APIs tab.  
The Develop page opens.
2. Click the title of the API definition that you want to work with.
3. Use the Policies view to add an element.

Note: You must specify a version for the policy. To prevent errors due to incompatible versions of a policy between the API and the gateway, API Connect will only publish an API to the gateway if the policy version in the API exactly matches the version that is present on the gateway.

- a. Select the Gateway tab, then click Policies in the navigation pane.
- b. Find the element that you want to add in the palette.
- c. Drag the element onto the canvas; dashed boxes are displayed. Drop the element in a dashed box to insert it into that position in the assembly.

Note:

- Elements are applied in order from the initial, unfilled, circle to the final, filled, circle.
  - Unless an Operation Switch element is used, the whole assembly applies to every operation in the API.
- d. Add and edit properties of the element by clicking the element and using the property sheet.  
For some elements, you can add and remove properties by clicking Object Properties and selecting the property from the drop-down list. For information about the properties of policy elements, see [API policies and logic constructs](#). For information about the properties of logic constructs, see [Logic Constructs](#).
  - e. Specify a version for the policy.  
To add a version, click the Source icon , and complete the `version` section of the policy YAML. For example:

```
execute:
- invoke:
 version: 2.2.0
 title: invoke
...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions. For more information about how to configure the policy OpenAPI YAML, see

[execute OpenAPI extension.](#)

- Optional: Repeat Steps 3.b to 3.e for any additional elements that you want to add.
- Click Save to save your changes.

## Results

---

You have added one or more elements to your assembly.

## Handling errors in the assembly

Use the catch section of the assembly to describe the handling of errors thrown during the assembly execution.

### About this task

---

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see [Editing an OpenAPI 3.0 API definition](#).






You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

The catch section of the assembly is used to implement an assembly in the instance that an error is thrown during the assembly execution. For example, the assembly could contain a throw element, the API caller could fail to authenticate, or a policy could fail to execute correctly. Each error can be handled with a different catch and each catch can handle multiple status errors.

### Procedure

---

To create a catch and include elements in it, complete the following steps:

- In the navigation pane, click  Develop, then select the APIs tab.  
The Develop page opens.
- Click the title of the API definition that you want to apply a catch to.
- Select the Gateway tab, then click Policies in the navigation pane.
- Set the Show catches toggle in the menu bar of the canvas to the Show position.
- Click Catches in the canvas, or a Catch icon  if one is displayed.  
The property sheet for the API's catches opens.
- To add a default catch that is executed when an otherwise uncaught error is thrown, click Add default catch.  
Note: If you have a default catch before another catch in precedence, the default catch will activate even when the other catch's error is thrown.
- To add a new catch, click Add catch.
- To specify which errors the catch applies to, type the name of a custom error and press Enter, or use the Search errors field to search for the appropriate error.
- Optional: To remove an error case from a catch, click the corresponding cross.
- Optional: To change the precedence of your catches, use the Move up  or Move down  icons.  
If an error case is handled by multiple catches, the catch at the beginning of the list is applied.
- To add an element to a catch, drag the element over the dashed, gray box that appears in the flow from the Catch icon  for the catch that you want to apply the element to.
- Click Save to save your changes.

### What to do next

---

If you added a catch for ConnectionError, SOAPError, or OperationError, you must add the same error to the Stop on error setting for the Invoke policy in your assembly. Otherwise if the error occurs during the execution of the Invoke policy, it is not caught, the policy execution is allowed to complete, and the assembly flow continues. For details on configuring an Invoke policy, see [Invoke](#).

For details of all the errors that can be returned by the assembly and are available to the catch function, see [Error cases supported by assembly catches](#).

## Specifying the gateway type for an API definition

An API definition is specific to one or other of the gateway types, DataPower® API Gateway or DataPower Gateway (v5 compatible). A default gateway type is set when you create an API definition, but you can edit the API configuration to specify a different gateway type.

### About this task

---

Note: This task relates only to an OpenAPI 2.0 API definition. An OpenAPI 3.0 API is supported only on the DataPower API Gateway.

DataPower Gateway (v5 compatible) has been available with IBM® API Connect for a number of years. The DataPower API Gateway is a new gateway that has been designed with APIs in mind, and with the same security focus as DataPower Gateway (v5 compatible).

For more information on how to choose which gateway type to use, see [API Connect gateway types](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You must specify which type of gateway each API uses. APIs can use only one type of gateway.

When you create a new API definition, or import an API definition that has no gateway type configured, the gateway type defaults according to the gateway types of the gateway services that are enabled in your Sandbox Catalog, as indicated in the following table.

Table 1. Default gateway type for a new API definition

Gateway Service enablement in Sandbox Catalog	Default gateway type for new API definition
No gateway services enabled	DataPower API Gateway
At least one gateway service of each type enabled	DataPower API Gateway
One or more gateway services only of type DataPower API Gateway enabled	DataPower API Gateway
One or more gateway services only of type DataPower Gateway (v5 compatible) enabled	DataPower Gateway (v5 compatible)


If you import an API definition that already has a gateway type configured, that gateway type is retained.

Note that when you modify your API definitions to use a specific gateway type, you must ensure that each policy and policy version in the API are supported by the gateway type. DataPower Gateway (v5 compatible) and DataPower API Gateway each support policies that the other gateway type does not. In some cases, the same policy is supported by both gateway types, but with a different version number.

For example, DataPower Gateway (v5 compatible) supports version **1.0.0** of the **invoke** policy, but DataPower API Gateway requires version **2.0.0**.

For information on policies, see [execute](#).

For information on policy versions, see the documentation for each individual policy. For example, to review the **invoke** policy, see [invoke](#).


At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Click the title of the API you want to update.
3. Select the Gateway tab, then click Gateway and portal settings.
4. Select the type:
  - datapower-gateway for the DataPower Gateway (v5 compatible)
  - datapower-api-gateway for the DataPower API Gateway.

Note: The Gateway Type selection is not applicable if the Enforced option is disabled, because an unenforced API is not managed on an API Connect gateway.

5. Click Save to retain the changes.
6. In the navigation pane, select Policies to ensure that your API uses policies that are supported by the gateway type you selected.

Policies in an existing API that are not supported by the new gateway type are grayed out and flagged with a red exclamation point: .

Some policies are supported by only one gateway type. Some policies are supported by both gateway types, but require a different version of the policy for each gateway type.

## Related tasks

- [Converting an API definition for deployment to the DataPower API Gateway](#)

## Converting an API definition for deployment to the DataPower API Gateway

IBM® API Connect provides two gateway types, DataPower® Gateway (v5 compatible) and DataPower API Gateway. If you have an API definition that was developed for the DataPower Gateway (v5 compatible) and you want to port it to the DataPower API Gateway, you must make changes to convert it before deployment.

### About this task

Note: This task relates only to an OpenAPI 2.0 API definition. An OpenAPI 3.0 API is supported only on the DataPower API Gateway.

DataPower Gateway (v5 compatible) has been available with IBM API Connect for a number of years. The DataPower API Gateway is a new gateway that has been designed with APIs in mind, and with the same security focus as DataPower Gateway (v5 compatible).

To convert an API for deployment to the DataPower API Gateway, you must, as a minimum, change the gateway type setting on the API definition, and ensure that each policy and policy version in the API are supported by the DataPower API Gateway; DataPower Gateway (v5 compatible) and DataPower API Gateway each support policies that the other gateway type does not. In some cases, the same policy is supported by both gateway types, but with a different version number.

In addition, depending on the specific policies in your API assembly, some policies might require modification to adapt them for use with the DataPower API Gateway.

The following procedure provides instructions for converting your API definition:

### Procedure

1. Set the gateway type for the API to DataPower API Gateway, and ensure policy type and version compatibility. For details, see [Specifying the gateway type for an API definition](#).
2. Refer to the [DataPower API Gateway porting notes](#) for detailed instructions on how to convert your API definition.

## Modifying a GraphQL schema

For a GraphQL API, an editor is provided for modifying the associated GraphQL schema.

### About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

## Procedure

---

To open the GraphQL schema editor for a GraphQL API, complete the following steps:

1. Open the API for editing, as described in [Editing an OpenAPI 2.0 API definition](#).
2. Select the GraphQL Schema tab.

## What to do next

---

For full details on how to use the editor, see [Using the GraphQL schema editor](#).

## Validating the OpenAPI YAML source

---

How to validate the OpenAPI YAML source when you're editing an API.

### About this task

---

When you're configuring an API in the API editor, you can validate the OpenAPI YAML source by clicking Validate. If any swagger parser validation errors are found, an error icon is displayed that you can then click to see the individual errors, and where they occur in the YAML. You can then fix the errors, save the updates, and continue to validate the OpenAPI YAML throughout the API configuration process.

Note: OpenAPI validation does not show gateway or policy errors that will stop your API from being staged or published. This type of validation occurs only when you try to stage or publish your API. For more information, see [Staging an API](#) or [Publishing an API](#).

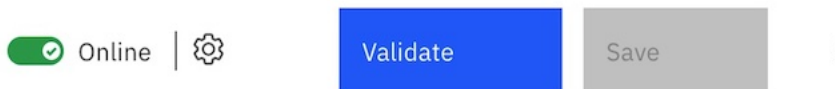
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

## Procedure

---


To validate the OpenAPI YAML source, complete the following steps.

1. Open the required API for editing, as described in [Editing an OpenAPI 2.0 API definition](#) or [Editing an OpenAPI 3.0 API definition](#).
2. Click Validate from the menu bar in the header.



3. If any swagger parser validation errors are found, an error icon is displayed on the header menu bar, with the number of OpenAPI errors found displayed next to the icon.

For example,  1.

4. You can then click the error icon to see a detailed list of the errors, and their location in the YAML. If you're in Form  view, you can click an individual error to navigate directly to the relevant API section in the form.
5. After you fix the error, click Save to save your updates, and then click Validate again.
6. When all of the errors are fixed, the validation returns a success message.

For example,  API has been validated.

## Related concepts

---

- [Testing an API](#)

## Related tasks

---

- [Activating an API](#)
- [Staging an API](#)

## Editing an OpenAPI 3.0 API definition

---

IBM® API Connect provides a form based editor that enables you to configure APIs that conform to the OpenAPI 3.0 specification.

### About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.



## Procedure

---

You can open an API for editing in either of the following ways:

- During the initial creation of an API, the API wizard guides you to enter the minimum configuration settings; on completion of the initial configuration, click Edit API.
- To open an existing API for editing, complete the following steps:
  1. In the navigation pane, click  Develop, then select the APIs tab.
  2. Click the title of the API version that you want to work with.
- [OpenAPI 3.0 support in IBM API Connect](#)  
IBM API Connect supports the OpenAPI 3.0 specification, with some limitations. The current implementation includes complete support for the Berlin Group NextGen PSD2 requirements.
- [Specifying API metadata](#)  
API metadata provides summary information about the API. You enter API metadata in the Info section of the API definition.
- [Defining servers for an API](#)  
Server definitions in an API provide information for connecting to target servers.
- [Enforcing security requirements on an API](#)  
To enforce security requirements on an API, you apply previously created security scheme components that define various aspects of API security configuration.
- [Specifying external documentation for an API](#)  
You can reference an external resource for extended documentation for an API.
- [Adding tags to an API](#)  
You can include tags in API definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.
- [Defining Paths for an API](#)  
A Path is a unit of a REST API that you can call. A Path comprises an HTTP verb and a URL path. By configuring the Path, you define how the API is exposed to your developers.
- [Defining components for an API](#)  
Components are reusable objects relating to various aspects of your API definition. A component has no effect on the API unless it is explicitly referenced from elsewhere in your API definition.
- [Specifying gateway and portal settings](#)  
Define general configuration settings for your API.
- [Defining target services for an API](#)  
A target service defines a web service that you want to use in your API definition.
- [Setting API properties](#)  
In addition to the pre-supplied API properties that you can use to control the behavior of API Connect policies, you can define your own API properties. The properties that you define can be referenced in your API definitions.
- [Configuring activity logging](#)  
You can configure your logging preferences for the API activity that is stored in Analytics, overriding the default activity logging behavior.
- [Including elements in your API assembly](#)  
An assembly is formed of elements that are applied to calls to and responses from operations in your API. Elements can be either policies or logic constructs.
- [Validating the OpenAPI YAML source](#)  
How to validate the OpenAPI YAML source when you're editing an API.

---

## OpenAPI 3.0 support in IBM API Connect

IBM® API Connect supports the OpenAPI 3.0 specification, with some limitations. The current implementation includes complete support for the Berlin Group NextGen PSD2 requirements.

### Overview

---

A Product can contain any combination of OpenAPI 2.0 and OpenAPI 3.0 APIs. When you publish a Product that contains an OpenAPI 3.0 API, that API is validated to ensure that it is syntactically correct, and that references to configuration resources and policies resolve correctly, in the same way that OpenAPI 2.0 APIs are validated.

You can also validate OpenAPI 3.0 APIs in your local file system by using the `apic validate` command provided by the developer toolkit CLI; for details, see [Validating the YAML or JSON definition of an API or Product](#).

If you retrieve an API object by using the developer toolkit CLI or the API Connect REST APIs, there is an `oai_version` property that defines which OpenAPI version the API represents.

There is no OpenAPI 3.0 API support with the DataPower® Gateway (v5 compatible); OpenAPI 3.0 API support is provided by the DataPower API Gateway **only**.

### Limitations

---

The limitations to the OpenAPI 3.0 support in IBM API Connect are as follows:

User interface limitations

- Some aspects of the OpenAPI 3.0 specification are not currently supported by the user interface. In such circumstances, use the OpenAPI source directly.
- Validation errors that are identified locally are reflected in the API editor header almost immediately. However, some validation errors can be identified only by the API Manager backend and are not reflected until the API is saved; this means that, on saving an API, some validation errors might appear or disappear.
- The user interface does not currently support referencing a request body component from the request body of an operation. If you want to reference a request body component, add the reference to the request body of the operation directly in the OpenAPI source:

```
requestBody:
 $ref: '#/components/requestBodies/request_body_component_name'
```

The reference is confirmed on the details page for the request body in the user interface however.

- If, in the API Designer user interface, you create and activate an API of the same name and version as one that has already been activated from the API Manager user interface, the Edit Rate Limit operation fails.

Limitations for APIs that are enforced by the DataPower API Gateway

- General limitations.
    - The **servers** array cannot contain more than one server.
    - The **url** entry in the **servers** array cannot contain **variables**.
    - **path** objects cannot contain server object overrides.
    - Wildcarding in response objects is not supported for error codes or success codes.
    - There is no support for converting a WSDL defined SOAP service into an OpenAPI 3.0 API.
  - Assembly policies.
    - All policies are supported.
    - JSON schema support is limited to Draft 4 and earlier. Therefore, the following policies are limited to those drafts:
      - **gatewayscript**
      - **set-variable**
      - **switch**
      - **json-to-xml**
      - **xml-to-json**
      - **parse**
      - **xslt**
      - **map**
      - **validate**
- For detailed information about the Draft 4 and earlier specifications, see <https://json-schema.org/specification-links.html#draft-4>.
- Security.
    - The OpenID Connect (OIDC) security scheme is not supported.
    - The use of cookies in an API key is not supported.

---

## Specifying API metadata

API metadata provides summary information about the API. You enter API metadata in the Info section of the API definition.



### About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand the General section, then click Info.
3. Provide the following information:
  - Title (required): The title of the API.
  - Name: The name was generated automatically when the API was created and cannot be changed. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see [apic draft-apis](#).
  - Version: The API version is specified when the API is initially created and cannot be changed thereafter. You can, however, create a new version of an existing API; see [Creating a new version of an API definition](#).
  - Description: An optional description of the API. You can use [CommonMark syntax](#) for rich text representation.
  - Terms of Service: A URL to the Terms of Service for the API.
  - Contact Name: The identifying name of the contact person/organization.
  - Contact URL: A URL pointing to the contact information.
  - Contact Email: The email address of the contact person/organization.
  - License Name: The license name used for the API.
  - License URL: A URL to the license used for the API.
4. Click Save when done.

---

## Defining servers for an API

Server definitions in an API provide information for connecting to target servers.

### About this task



---

Note:


- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can define more than one server but only the first is used by API Connect.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand General then, if there are already one or more servers defined, expand Servers.
3. To create a new server definition, click the add icon  alongside Servers in the navigation pane. To edit an existing server definition, click the server URL in the navigation pane.
4. Provide the following information:
  - Server URL (required): The specified URL is used to determine the full URL endpoint for the calling the API, taking into account any vanity endpoint configuration in the Catalog in which the API is published. For an API that is enforced by the DataPower API Gateway, the value entered here is interpreted as the basepath so you would typically provide only the basepath value; for example:
 

```
/my_basepath
```

For full details on how the server URL is used to determine the full URL endpoint, see [Configuring vanity endpoints for a Catalog](#).
  - Server Description: An optional description of the host designated by the URL. You can use [CommonMark syntax](#) for rich text representation.
  - Server Variables (available when editing an existing server definition): A server variable defines a map between a variable name and its value. The value is used for substitution in the server's URL template.
    - To add a new server variable, click Add. To edit an existing server variable, click the variable name.
    - Provide the following information:
      - Server Variable Name (required).
      - Default value (required).
      - An optional rich text description. You can use [CommonMark syntax](#) for rich text representation.
      - One or more Enum Value entries (available when editing an existing server variable). Enum values specify an enumeration of string values to be used if the substitution options are from a limited set. To add a new Enum value, click Add, enter the value, then click Create. To edit an existing Enum value, click the Enum value.
    - If you are creating a new server variable, click Create. The server variable details are displayed for further editing.
    - If required, use the breadcrumb trail to return to your server definition for further editing.
5. If you are creating a new server definition, click Create.  
The server definition details are displayed for further editing as described in step 4.
6. Click Save when done.

## Enforcing security requirements on an API

To enforce security requirements on an API, you apply previously created security scheme components that define various aspects of API security configuration.

### About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).



You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

For details on how to create and configure security scheme components, see [Defining security scheme components](#).

The following restrictions exist when you apply security schemes to an API:

- You cannot apply more than two API key security schemes to an API.
- If you apply an API key security scheme for client secret, you must also apply an API key security scheme for client ID.
- If you require the application developer to supply both client ID and client secret, you must apply two separate API key security schemes.
- You can have at most one API key scheme of type client ID, regardless of whether the client ID is sent in the request header or as a query parameter.
- You can have at most one API key scheme of type client secret, regardless of whether the client secret is sent in the request header or as a query parameter.
- You cannot apply more than one basic security scheme to an API. If you apply a basic security scheme, you cannot also apply an OAuth security scheme.
- You can apply at most one OAuth security scheme to an API.

A security requirement specifies one or more security scheme components whose conditions must all be satisfied for the API to be called successfully. You can define multiple security requirements; in this case, an application can call your API if it satisfies any of the security requirements you have defined.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).

2. Expand General.



3. To create a new security requirement for the API, complete the following steps:

- Click the add icon  alongside Security in the navigation pane.
- Select the security schemes that you want to include in this security requirement. The security schemes listed are those that have been defined in security scheme components; see [Defining security scheme components](#).

If a selected security scheme is of type OAuth2, select the required scopes; the scopes available for selection are those that were specified in the security scheme component; for more information, see [Defining OAuth2 security scheme components](#).

If you are applying the OAuth2 security scheme to an API that is enforced by the DataPower API Gateway, you only need select any scopes if Advanced scope check after token generation is not enabled in the native OAuth provider associated with the security scheme. If a default scope has been set in the native OAuth provider and the API request doesn't contain any scope, the default scope is used; for more information, see [Configuring scopes for a native OAuth provider](#).

Note: The following additional requirement applies to security schemes that will be used with an OAuth third party provider. If you select an OAuth security scheme for protecting a consumer API, you must also include an API key security scheme, as the `X-IBM-Client-Id` or `client_id` must be included in the security credentials so that the correct Plan configuration settings can be enforced.

- Click Create. The security scheme selections are shown; you can change them again before saving.
  - Click Submit when done.
4. To modify an existing security requirement, complete the following steps:
- Click Security in the navigation pane. All previously defined security requirements are listed; the security schemes included in each security requirement are shown.
  - To change the security schemes for a security requirement, click the edit icon  alongside the required security requirement, then change your security requirement selections as required.
  - Click Submit when done, then click Save.
  - To delete a security requirement, click the appropriate delete icon , click Delete to confirm, then click Save.
  - To disable security for the API, clear the Require one of the following Security Requirements check box, then click Save.

## What to do next

---

For more information on LDAP and Authentication URL, see [LDAP authentication](#) and [Authentication URL user registry](#).

- [Enabling CORS support for an API](#)

You can enable cross-origin resource sharing (CORS) support for your API. CORS allows embedded scripts in a web page to call the API across domain boundaries.

---

## Enabling CORS support for an API

You can enable cross-origin resource sharing (CORS) support for your API. CORS allows embedded scripts in a web page to call the API across domain boundaries.

## Before you begin

---

This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).



## About this task

---

Note:

- When CORS is enabled, the API Gateway runs the `cors` preflow policy to handle all CORS requests that are made to the API.
- When CORS is enabled and a preflight request is received, only the following API actions are performed:
  - The `cors` preflow policy configures the appropriate response headers.
  - The response headers are set.
- When a preflight request is received, the `request.attributes.isCORSPreflight` flag is set to `true`.
- For all preflight requests, the `security` and `client-identification` preflow policies are always skipped, whether CORS is enabled or not enabled.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

To enable CORS support for an API, complete the following steps:

- Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
- Select the Gateway tab, expand Gateway and portal settings, then click CORS. The CORS page opens.
- Select Enable CORS.
- Optional: Configure a CORS policy.

Should you create a CORS policy? Review the following considerations:

- A CORS policy is an optional part of an API definition. If an API definition has no CORS policy but CORS is enabled, then CORS requests will be accepted from all Origins.  
If you want to accept CORS requests from all Origins, then enable CORS but do not add a CORS policy to the API definition.
- If you create a CORS policy, then CORS requests will only be accepted from Origins that are explicitly listed in the CORS rules contained in the CORS policy. CORS requests from any other Origins will be rejected.  
If you want to accept CORS requests from only a limited number of Origins (and might also want to configure the Access-Control-Allow-Credentials and Access-Control-Expose-Headers response headers), then you should enable CORS and create a CORS policy. Only those Origins explicitly listed in the allow-origin field of the CORS rules in the CORS policy will be accepted; CORS requests from any Origins not listed will be rejected.

- To configure a new CORS policy, complete the following steps:
  - Alongside CORS Policy, click Add.
  - To include the header **Access-Control-Allow-Credentials: true** in a response, select Allow Credentials. The **Access-Control-Allow-Credentials** response header tells browsers whether to expose the response to frontend JavaScript code when the request's credentials mode (**Request.credentials**) is set to **include**.
  - To append one or more of the following values to the **Access-Control-Expose-Headers** response header, select Expose headers, and select from the following options:
    - Predefined - The predefined value of the Gateway. This option is selected by default.
    - Backend - The value of **Access-Control-Expose-Headers** from the backend response.
    - Custom - A custom string.
  - Click Create.
  - Alongside Allowed Origins click Add.
  - Enter the origin URL, then click Create. This setting indicates that the response can be shared with requesting code from the specified origin.
- To modify an existing CORS policy, click its Allowed Origins entry on the CORS page. You can then modify individual origin URLs, add further origins, and change the Allow Credentials setting.

#### Example

The following example has three rules:

- Accept an Origin header of **https://example.com**, and return **Access-Control-Allow-Credentials: true** in the CORS response.
- Accept an Origin header of **http://domain.com**. No **Access-Control-Allow-Credentials** header will be returned in the response.
- Accept any of the following Origin headers:
  - **http://example2.com**
  - **http://example3.com**
  - **http://example4.com**

No **Access-Control-Allow-Credentials** header will be returned in the response.

```
cors:
 enabled: true
 policy:
 -
 allow-origin:
 - 'https://example.com'
 allow-credentials: true
 -
 allow-origin:
 - 'http://domain.com'
 -
 allow-origin:
 - 'http://example2.com'
 - 'http://example3.com'
 - 'http://example4.com'
```

5. Click Save to save your changes.
6. Optional: To implement your own CORS solution using custom OPTIONS operations, complete the following steps:
  - a. Add the following headers to your HTTP responses:

```
Access-Control-Allow-Origin: https://<portalhostname>
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
```

Where **<portalhostname>** is your Developer Portal host name.

- b. Optional: You can proxy your API through API Connect as an enforced invoke API so that CORS is handled automatically.

Important:

- If you implement your own CORS solution, you **must** disable the CORS option described in step 3
- CORS preflight requests are sent by using the HTTP **OPTIONS** method. Therefore, if you require these requests to be handled by the API Connect gateway then you must enable the **OPTIONS** method for all APIs that will handle preflight requests; see [Defining Paths for a REST API](#).
- **OPTIONS** requests are counted as API calls against any configured rate limit. Note that you can apply rate limits to individual operations; see [Defining rate limits for an API operation](#).

---

## Specifying external documentation for an API

You can reference an external resource for extended documentation for an API.



### About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand the General section, then click External Documentation.
3. Provide the following information:

- URL (required): The URL for the target documentation.
  - Description: An optional description of the target documentation. You can use [CommonMark syntax](#) for rich text representation.
4. Click Save when done.

---

## Adding tags to an API

You can include tags in API definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.



---

### About this task

Note:


- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand the General then, if there already one or more tags defined, expand Tags.
3. To add a new tag, click the add icon  alongside Tags in the navigation pane. To edit an existing tag, click the tag name in the navigation pane.
4. Provide the following information:
  - Name (required): The name of the tag.
  - Description: An optional description of the tag. You can use [CommonMark syntax](#) for rich text representation.
  - External Documentation URL (available when editing an existing tag): The URL for target documentation that describes this tag.
  - External Documentation Description (available when editing an existing tag): An optional description of the target documentation.
5. If you are creating a new tag, click Create.  
The tag details are displayed for further editing as described in step 4.
6. Click Save when done.

---

## Defining Paths for an API

A Path is a unit of a REST API that you can call. A Path comprises an HTTP verb and a URL path. By configuring the Path, you define how the API is exposed to your developers.

---



### About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).


You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

The Path is appended to the server URL to form the full URL. For details on configuring servers, see [Defining servers for an API](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. If there are already one or more Paths defined, expand Paths.
3. To create a new Path, click the add icon  alongside Paths in the navigation pane. To edit an existing Path, click the Path in the navigation pane.
4. Provide the following information:
  - Path: A relative path to an individual endpoint.  
The path is appended to the base path to construct the full URI to access the APIs. The path must start, but not end, with the / character. A parameter at the end of the path can contain a qualifier to match one or more path levels.  
  
If you specify just the name of the parameter, then one level of that path is matched. If you want to allow for multiple levels of the path, you can prefix the parameter with one of the following qualifiers:
    - \* to indicate 0 or more occurrences
    - + to indicate 1 or more occurrencesThe + and \* qualifiers can only be used at the end of the path.  
  
For example, the path:

```
/petstore/{type}/{*category}
```

matches the following paths, where only one type value is matched, but all (0 or more) categories are matched:

```
/petstore/cats
/petstore/cats/supplies
/petstore/cats/supplies/health
/petstore/cats/supplies/health/medicines
/petstore/cats/supplies/health/medicines/a/b/c
```

- Ref (available when editing an existing Path): A reference to a Path that is defined elsewhere, either in this OpenAPI definition, or in an external file. A reference enables reuse of a set of operations already defined for a Path in this API, or in a file that is either a complete API definition or an OpenAPI fragment.

The reference uses JSON pointer notation. For example, a reference to a Path called `/mypath` in this API would have the following format:

```
#/paths/~1mypath
```

The `~1` characters are used to escape the `/` character, which is being used literally in the path name rather than as a special character.

A reference to a path in an external file called `path_fragments.yaml` would have the following format:

```
file://path_fragments.yaml/paths/~1mypath
```

- Summary: An optional summary, intended to apply to all operations in this path.
- Description: An optional description, intended to apply to all operations in this path. You can use [CommonMark syntax](#) for rich text representation.

5. If you are creating a new Path, click Create.

The Path details are displayed for further editing as described in [step 4](#).

6. Click Save when done.

- **Defining servers for a Path**

Server definitions for a Path provide alternative target servers when calling the API at that Path.

- **Defining parameters for a Path**

Path parameters can specify variable elements of a URL path, query parameters, headers, or a request body. The parameters apply to all the operations defined under the path. A Path parameter can be overridden at the operation level but cannot be removed there.

- **Defining operations for a Path**

Operations on a Path defines the mechanisms available for interacting with the API when it is called using that Path. An operation can be of various pre-defined types; for example, a POST operation is most commonly used for creating new resources, while a GET operation is used to retrieve a resource.

---

## Defining servers for a Path

Server definitions for a Path provide alternative target servers when calling the API at that Path.

---

### About this task



Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.


Note: It is unlikely that you will need to define servers for a Path because it applies only if the API operations are distributed across different gateway endpoints, and such a configuration is possible only if you are hosting your API operations on your own runtime endpoints outside of API Connect. For an API that is published to the DataPower API Gateway, all API operations are invoked on the same gateway service.

A server defined for a Path overrides any server defined for the parent API. You can define more than one server but only the first is used by API Connect.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Paths.
3. Expand the required Path, then, if there already one or more servers defined for the Path, expand Servers.
4. To create a new server definition for the Path, click the add icon  alongside Servers under the Path in the navigation pane. To edit an existing server definition, click the server URL under the Path in the navigation pane.
5. Provide the following information:
  - Server URL (required): The specified URL is used to determine the full URL endpoint for the calling the API, taking into account any vanity endpoint configuration in the Catalog in which the API is published. For an API that is enforced by the DataPower API Gateway, the value entered here is interpreted as the basepath so you would typically provide only the basepath value; for example:

```
/my_basepath
```

For full details on how the server URL is used to determine the full URL endpoint, see [Configuring vanity endpoints for a Catalog](#).

- Server Description: An optional description of the host designated by the URL. You can use [CommonMark syntax](#) for rich text representation.
- Server Variables (available when editing an existing server definition): A server variable defines a map between a variable name and its value. The value is used for substitution in the server's URL template.
  - To add a new server variable, click Add. To edit an existing server variable, click the variable name.
  - Provide the following information:
    - Server Variable Name (required).

- Default value (required).
  - An optional rich text description. You can use [CommonMark syntax](#) for rich text representation.
  - One or more Enum Value entries (available when editing an existing server variable). Enum values specify an enumeration of string values to be used if the substitution options are from a limited set. To add a new Enum value, click Add, enter the value, then click Create. To edit an existing Enum value, click the Enum value.
    - If you are creating a new server variable, click Create. The server variable details are displayed for further editing.
    - If required, use the breadcrumb trail to return to your server definition for further editing.
6. If you are creating a new server definition, click Create.  
The path details are displayed for further editing as described in [5](#).
7. Click Save when done.

---

## Defining parameters for a Path

Path parameters can specify variable elements of a URL path, query parameters, headers, or a request body. The parameters apply to all the operations defined under the path. A Path parameter can be overridden at the operation level but cannot be removed there.

---

### About this task

Note:



- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A parameter is similar to a header, with the following differences:


- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **header**.
- A parameter can have various format styles, the style of a header is always **simple**.

For more information on headers, see [Creating a header](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Paths.
3. Expand the required Path, then, if there already one or more parameters defined, expand Parameters.
4. You can either create a new parameter, or edit any existing parameter.
  - To create a parameter, click the add icon  alongside Parameters, then refer to [Creating a parameter](#).
  - To edit an existing parameter, click the parameter name in the navigation pane, then refer to [Editing a parameter](#).

---

## Defining operations for a Path

Operations on a Path defines the mechanisms available for interacting with the API when it is called using that Path. An operation can be of various pre-defined types; for example, a POST operation is most commonly used for creating new resources, while a GET operation is used to retrieve a resource.



---

### About this task

Note:


- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Paths.
3. Expand the required Path, then, if there are one or more operations, expand Operations.
4. To create an operation, click the add icon  alongside Operations under the Path in the navigation pane. To edit an existing operation, click the operation name under the Path in the navigation pane.
5. Provide the following information:
  - Verb: Select the operation type from the following options:
    - get



- post
  - put
  - delete
  - options
  - head
  - patch
  - trace
  - Summary: A summary of what the operation does.
  - Description: A description of the operation behavior. You can use [CommonMark syntax](#) for rich text representation.
  - Operation ID: Unique string used to identify the operation. The id must be unique among all operations described in the API.
  - Deprecated: Declares this operation to be deprecated. Consumers should refrain from using the declared operation.
6. If you are creating a Path operation, click Create.
  7. Click Save when done.

- [Adding tags to an operation](#)  
You can include tags in API operation definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.
- [Specifying external documentation for an operation](#)  
You can referencing an external resource for extended documentation for an API operation.
- [Defining parameters for an operation](#)  
Operation parameters can, for the URL path associated with that specific operation, define variable elements of the URL path, query parameters, headers, or body parameters.
- [Defining the request body for an operation](#)  
A request body defines the structure of the body of an API request.
- [Defining responses for an operation](#)  
Responses define the HTTP status code and data returned in a response body and headers.
- [Enforcing security requirements on an operation](#)  
To enforce security requirements on an API operation, you apply previously created security scheme components that define various aspects of API security configuration.
- [Defining servers for an operation](#)  
Server definitions for an API operation provide alternative target servers when calling that operation.

---

## Adding tags to an operation

You can include tags in API operation definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.


### About this task

---

Note:


- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Paths, then expand the required Path.
3. Expand Operations, then expand the required operation.
4. To add a new operation tag, click the add icon  alongside the Tags entry for the operation in the navigation pane. To edit an existing tag, expand Tags then click the tag name in the navigation pane.
5. Provide the following information:
  - Name (required): The name of the tag.
6. If you are creating a new operation tag, click Create.  
The tag details are displayed for further editing as described in step 5.
7. Click Save when done.

---

## Specifying external documentation for an operation

You can referencing an external resource for extended documentation for an API operation.



### About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Paths, then expand the required Path.
3. Expand Operations, expand the required operation, then click External Documentation.
4. Provide the following information:
  - URL (required): The URL for the target documentation.
  - Description: An optional description of the target documentation. You can use [CommonMark syntax](#) for rich text representation.
5. Click Save when done.

---

## Defining parameters for an operation

Operation parameters can, for the URL path associated with that specific operation, define variable elements of the URL path, query parameters, headers, or body parameters.

---

## About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).



You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

An operation parameter overrides a parameter of the same name defined on the parent Path; see [Defining parameters for a Path](#).

A parameter is similar to a header, with the following differences:


- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **header**.
- A parameter can have various format styles, the style of a header is always **simple**.

For more information on headers, see [Creating a header](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Paths, then expand the required Path.
3. Expand Operations, then expand the required operation.
4. You can either create a new operation parameter, or edit any existing parameter.
  - To create an operation parameter, click the add icon  alongside Parameter, then refer to [Creating a parameter](#).
  - To edit an existing operation parameter, click the parameter name in the navigation pane, then refer to [Editing a parameter](#).

---

## Defining the request body for an operation

A request body defines the structure of the body of an API request.



---

## About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Note: The user interface doesn't currently support referencing a request body component from an operation. However, you can add the reference directly to the OpenAPI YAML source for your API definition, for example:

```
paths:
 /mypath:
 get:
```

```
requestBody:
 $ref: '#/components/requestBodies/my_requestbody_component'
```

For information on request body components, see [Defining request body components](#).

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Paths, then expand the required Path.
3. Expand Operations, expand the required operation, then click requestBody.
4. Provide the following information:
  - Description: An optional description of the request body. This could contain examples of use. You can use [CommonMark syntax](#) for rich text representation.
  - Content: A content definition describes the content of the request body.  
To define a new content definition for the request body, click Add, then refer to [Creating a content definition](#).  
  
To edit an existing content definition, click the content definition name, then refer to [Editing a content definition](#).
  - Required: Determines if the request body is required in the request.
5. Click Save when done.

## Defining responses for an operation


Responses define the HTTP status code and data returned in a response body and headers.

### About this task


Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Paths, then expand the required Path.
3. Expand Operations, expand the required operation, then, if there are any existing responses, expand Responses.
4. You can either create a new response, or edit any existing response.
  - To create a response, click the add icon  alongside Responses, then refer to [Creating a response](#).
  - To edit an existing response, click the response name in the navigation pane, then refer to [Editing a response](#).

## Enforcing security requirements on an operation

To enforce security requirements on an API operation, you apply previously created security scheme components that define various aspects of API security configuration.

### About this task

Note:



- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.


For details on how to create and configure security scheme components, see [Defining security scheme components](#).

A security requirement specifies one or more security scheme components whose conditions must all be satisfied for the API operation to be called successfully. You can define multiple security requirements; in this case, an application can call your API operation if it satisfies any of the security requirements you have defined.

Any security requirements that you define for an operation completely override any security requirements defined on the parent API. If you do not define any security requirements for an operation, or you delete all security requirements from an operation, the operation inherits the security requirements defined on the parent API. For more information, see [Enforcing security requirements on an API](#).



At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Paths, then expand the required Path.
3. Expand Operations, then expand the required operation.
4. To create a new security requirement for the operation, complete the following steps:
  - a. Click the add icon  alongside the Security Requirements entry for the operation in the navigation pane.
  - b. Select the security schemes that you want to include in this security requirement. The security schemes listed are those that have been defined in security scheme components; see [Defining security scheme components](#).  
If a selected security scheme is of type OAuth2, select the required scopes; the scopes available for selection are those that were specified in the security scheme component; for more information, see [Defining OAuth2 security scheme components](#).

If you are applying the OAuth2 security scheme to an API that is enforced by the DataPower API Gateway, you only need select any scopes if Advanced scope check after token generation is not enabled in the native OAuth provider associated with the security scheme. If a default scope has been set in the native OAuth provider and the API request doesn't contain any scope, the default scope is used; for more information, see [Configuring scopes for a native OAuth provider](#).

Note: The following additional requirement applies to security schemes that will be used with an OAuth third party provider. If you select an OAuth security scheme for protecting a consumer API, you must also include an API key security scheme, as the `X-IBM-Client-Id` or `client_id` must be included in the security credentials so that the correct Plan configuration settings can be enforced.

- c. Click Create. The security scheme selections are shown; you can change them again before saving.
  - d. Click Submit when done.
5. To modify an existing security requirement, complete the following steps:
    - a. Click the Security entry for the operation in the navigation pane. All previously defined security requirements are listed; the security schemes included in each security requirement are shown.
    - b. To change the security schemes for a security requirement, click the edit icon  alongside the required security requirement, then change your security requirement selections as required.
    - c. Click Submit when done, then click Save.
    - d. To delete a security requirement, click the appropriate delete icon , click Delete to confirm, then click Save.
    - e. To disable security for the operation, clear the Require one of the following Security Requirements check box, then click Save.

Note: These settings completely override any security requirements defined on the parent API; see [Enforcing security requirements on an API](#).

## Defining servers for an operation

Server definitions for an API operation provide alternative target servers when calling that operation.



### About this task

Note:


- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A server defined for an operation overrides any server defined for the parent API or Path. You can define more than one server but only the first is used by API Connect.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Paths, then expand the required Path.
3. Expand Operations, expand the required operation, then, if there already one or more servers defined for the operation, expand Servers.
4. To create a new server definition for the operation, click the add icon  alongside Servers under the operation in the navigation pane. To edit an existing server definition, click the server URL under the operation in the navigation pane.
5. Provide the following information:
  - Server URL (required): The specified URL is used to determine the full URL endpoint for the calling the API, taking into account any vanity endpoint configuration in the Catalog in which the API is published. For an API that is enforced by the DataPower API Gateway, the value entered here is interpreted as the basepath so you would typically provide only the basepath value; for example:

```
/my_basepath
```

For full details on how the server URL is used to determine the full URL endpoint, see [Configuring vanity endpoints for a Catalog](#).

- Server Description: An optional description of the host designated by the URL. You can use [CommonMark syntax](#) for rich text representation.
- Server Variables (available when editing an existing server definition): A server variable defines a map between a variable name and its value. The value is used for substitution in the server's URL template.
  - To add a new server variable, click Add. To edit an existing server variable, click the variable name.
  - Provide the following information:
    - Server Variable Name (required).
    - Default value (required).
    - An optional rich text description. You can use [CommonMark syntax](#) for rich text representation.

- One or more Enum Value entries (available when editing an existing server variable). Enum values specify an enumeration of string values to be used if the substitution options are from a limited set. To add a new Enum value, click Add, enter the value, then click Create. To edit an existing Enum value, click the Enum value.
  - If you are creating a new server variable, click Create. The server variable details are displayed for further editing.
  - If required, use the breadcrumb trail to return to your server definition for further editing.
6. If you are creating a new server definition, click Create.  
The path details are displayed for further editing as described in [5](#).
7. Click Save when done.

---

## Defining components for an API

Components are reusable objects relating to various aspects of your API definition. A component has no effect on the API unless it is explicitly referenced from elsewhere in your API definition.

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

The following subtopics detail the various types of API component:

- [Defining schema components](#)  
Schema components define reusable schemas that provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation.
- [Defining response components](#)  
Response components define reusable response objects that specify the HTTP status code and data returned in a response body and headers.
- [Defining parameter components](#)  
Parameter components define reusable parameters that can specify variable elements of a URL path, query parameters, headers, or a request body.
- [Defining example components](#)  
An example component is a reusable example that you can add to an API to make the specification of your API clearer.
- [Defining request body components](#)  
A request body component is a reusable definition of the structure of the body of an API request.
- [Defining header components](#)  
A header component defines a reusable custom header that can be sent in an API request or returned in an API response.
- [Defining security scheme components](#)  
A security scheme component specifies all the settings for a particular aspect of API security; for example, the user registry that you use to authenticate access to the API.
- [Defining link components](#)  
A link component is a reusable link definition that can be used to describe how various values returned by one operation can be used as input for other operations. In this way, links provide a known relationship and traversal mechanism between the operations.

---

## Defining schema components

Schema components define reusable schemas that provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation.

---

### About this task



Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To make use of a schema component, you reference it from elsewhere in your API definition. You can reference a schema from the following locations:


- An API Path parameter; see [Defining parameters for a Path](#).
- An API operation parameter; see [Defining parameters for an operation](#).
- The request body of an API operation; see [Defining the request body for an operation](#).
- An API operation response; see [Defining responses for an operation](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Components then, if there are already one or more schemas components defined, expand Schemas.
3. You can create a new schema component, or edit any existing schema component.

- To create a schema component, click the add icon  alongside Schemas, then refer to [Creating a schema](#).
- To edit an existing schema component, click the schema component name in the navigation pane, then refer to [Editing a schema](#).
- **Creating a schema**  
Schemas provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation. You can create schemas in various places in your API definition.
- **Editing a schema**  
Schemas provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation. You can edit schemas that have been previously created in various places in your API definition.

## Creating a schema

Schemas provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation. You can create schemas in various places in your API definition.

### Before you begin

Note: Schemas are compiled before they are used for validation. Because the compilation process is longer than the validation process, the compiled schema artifacts are stored in a cache. The limited capacity of the cache can cause older entries to be evicted from the cache when newer entries are added. Schemas whose artifacts have been evicted from the cache must be recompiled, which can cause significant delays in validation.

Launch the schema creation window. For details of the areas in your API definition from where you can create a schema, see the following topics:



- [Editing a content definition](#)
- [Editing a header](#)
- [Editing a parameter](#)
- [Defining schema components](#)

### About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

1. Create the schema; you can either create a schema from scratch, or you can reference a schema that is defined in a schema component.
  - To create the schema from scratch, provide the following information on the Definition tab:
    - Schema Name: If you are creating a schema component, this name defines a key that enables this schema to be referenced from elsewhere in the API definition; the reference has the following format:  
**#/components/schemas/Name**
    - Title: The schema title.
    - Type: The schema data type; select one of the following:
      - array
      - boolean
      - integer
      - number
      - object
      - string
    - Description: An optional description of the schema. You can use [CommonMark syntax](#) for rich text representation.
    - Select the following options as required:
      - Read Only: A property associated with this schema can be sent in a response but should not be sent in a request.
      - Write Only: A property associated with this schema can be sent in a request but should not be sent in a response.
      - Deprecated: The schema is deprecated and should be transitioned out of usage.
  - To reference a schema that is defined in another schema component, provide the following information on the Reference tab:
    - Schema Name: If you are creating a schema component, this name defines a key that enables this schema to be referenced from elsewhere in the API definition; the reference has the following format:  
**#/components/schemas/Name**
    - Reference from: Select a reference to a schema component.
2. Click Create.  
The schema details are displayed for further editing; see [Editing a schema](#).

## Editing a schema

Schemas provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation. You can edit schemas that have been previously created in various places in your API definition.

## Before you begin

Note: Schemas are compiled before they are used for validation. Because the compilation process is longer than the validation process, the compiled schema artifacts are stored in a cache. The limited capacity of the cache can cause older entries to be evicted from the cache when newer entries are added. Schemas whose artifacts have been evicted from the cache must be recompiled, which can cause significant delays in validation.

Open the details form for a schema. For details of the areas in your API definition from where you can edit a schema, see the following topics:



- [Editing a content definition](#)
- [Editing a header](#)
- [Editing a parameter](#)
- [Defining schema components](#)

## About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure


You can configure the following schema settings:

- Schema Name: Available if you are editing a schema component, this name defines a key that enables this schema to be referenced from elsewhere in the API definition; the reference has the following format:

`#/components/schemas/Name`

To change the name, click Update, change the value in the field, then click Save.

- Title: The schema title.
- Type: The schema data type; select one of the following:
  - array
  - boolean
  - integer
  - number
  - object
  - string

Note: If you change the type of an existing schema, any existing settings that are not relevant to the new type are still retained in the OpenAPI source even though they are no longer displayed in the user interface. You should therefore avoid porting a schema from one type to another but instead either create a new schema or modify the OpenAPI source directly; to open the source editor, click the Source icon .

- Additional schema settings dependent on the selected Type, as follows:
  - array:
    - Max Items: The maximum allowed number of items in an array instance.
    - Min Items: The minimum allowed number of items in an array instance.
    - Unique Items: If selected, all items in an array instance must be unique.
  - boolean: no type specific settings.
  - integer:
    - Format: An optional data type modifier. For more information, see [Data Types](#) in the [OpenAPI Specification](#).
    - Multiple Of: Division of an integer instance by this value must result in an integer.
    - Maximum: An integer instance must be less than or equal to this value.
    - Exclusive Maximum: An integer instance must be strictly less than this value.
    - Minimum: An integer instance must be greater than or equal to this value.
    - Exclusive Minimum: An integer instance must be strictly greater than this value.
  - number:
    - Format: An optional data type modifier. For more information, see [Data Types](#) in the [OpenAPI Specification](#).
    - Multiple Of: Division of a number instance by this value must result in an integer.
    - Maximum: A number instance must be less than or equal to this value.
    - Exclusive Maximum: A number instance must be strictly less than this value.
    - Minimum: A number instance must be greater than or equal to this value.
    - Exclusive Minimum: A number instance must be strictly greater than this value.
  - object:
    - Properties: To add properties to the schema, click Add alongside the Properties section heading. You configure a property by defining its schema with the schema editor.  
To create the property, refer to [Creating a schema](#).

To further configure the property, follow the steps on this page.

By defining properties, you can create more complex data structures. For example, you could define an "Address" schema that has the following properties:

PROPERTY NAME	PROPERTY TYPE
---------------	---------------

PROPERTY NAME	PROPERTY TYPE
street1	string
street2	string
city	string
state	string
zip_code	string

and then define a "BankBranch" schema that has the following properties, one of which is a reference to the Address schema:

PROPERTY NAME	PROPERTY TYPE
address	address
type	string
id	string

- Max Properties: The number of properties in an object instance must be less than or equal to this value.
- Min Properties: The number of properties in an object instance must be greater than or equal to this value.
- Required Properties: An array of property names; every property name in the array must be the name of a property in an object instance. To add a required property name, click Add alongside the Required Properties section heading, enter the name, then click Create.
  - string:
    - Format: An optional data type modifier. For more information, see [Data Types](#) in the [OpenAPI Specification](#).
    - Pattern: A regular expression; a string instance must successfully match the expression.
    - Max Length: The maximum allowed number of characters in a string instance.
    - Min Length: The minimum allowed number of characters in a string instance.
    - Enum: An array of string values; a string instance must match one of the array values. To add an enum value, click Add, enter the value, then click Create.
- Description: An optional description of the schema. You can use [CommonMark syntax](#) for rich text representation.
- General settings; select the required options:
  - Read Only: A property associated with this schema can be returned in a response but should not be included in a request. For example, properties whose values are system generated identifiers will be defined as Read Only because application clients wouldn't be able to set these.
  - Write Only: A property associated with this schema can be included in a request but should not be returned in a response. For example, password properties are usually defined as Write Only because application clients would supply these when creating a user object but shouldn't be able see them when retrieving the object.
  - Deprecated: The schema is deprecated and should be transitioned out of usage.
- Example Value: An example value. Whatever you enter here is displayed as-is in the Developer Portal.
- External Documentation: An external resource for extended documentation; provide the following information:
  - URL (required): The URL for the target documentation.
  - Description: An optional description of the target documentation. You can use [CommonMark syntax](#) for rich text representation.
- Xml: Metadata that allows for more fine-tuned XML model definitions; provide the following information:
  - Name: Replaces the name of the element or attribute used for the described schema property.
  - Namespace: The URL of the namespace definition.
  - Prefix: The prefix to be used for the Name.
  - Attribute: Specifies whether the property definition translates to an attribute instead of an element.
  - Wrapped: For an array definition, Specifies whether the array is wrapped (for example, `<books><book/></books>`) or unwrapped (`<book/></book/>`).
- Click Save when done.

## What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

## Defining response components

Response components define reusable response objects that specify the HTTP status code and data returned in a response body and headers.



## About this task

Note:


- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To make use of a response component, you must reference it from elsewhere in your API definition.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Components then, if there are already one or more response components defined, expand Responses.
3. You can create a new response component, or edit any existing response component.
  - To create a response component, click the add icon  alongside Responses, then refer to [Creating a response](#).
  - To edit an existing response component, click the response component name in the navigation pane, then refer to [Editing a response](#).



- [Creating a response](#)  
Responses define the HTTP status code and data returned in a response body and headers. You can create responses for the Path operations in your API definition.
- [Editing a response](#)  
Responses define the HTTP status code and data returned in a response body and headers. You can edit responses that have been previously created for Path operations in your API definition.
- [Creating a content definition](#)  
A content definition describes the content of an API request, response, or parameter. You can create content definitions in various places in your API definition.
- [Editing a content definition](#)  
A content definition describes the content of an API request, response, or parameter. You can edit content definitions that have been previously created in various places in your API definition.
- [Creating an encoding definition](#)  
An encoding definition defines a map between a specific schema property and its encoding information.
- [Editing an encoding definition](#)  
An encoding definition defines a map between a specific schema property and its encoding information

---

## Creating a response

Responses define the HTTP status code and data returned in a response body and headers. You can create responses for the Path operations in your API definition.

### Before you begin

Launch the response creation window. For details of the areas in your API definition from where you can create a response, see the following topics:



- [Defining responses for an operation](#)
- [Defining response components](#)

### About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

1. Create the response; you can either create a response from scratch, or you can reference a response that is defined in a response component.
  - To create the response from scratch, provide the following information on the Definition tab:
    - Response name: A name for the response.  
If you are creating an operation response, this name is the HTTP status code.  
  
If you are creating a response component, this name defines a key that enables this response to be referenced from elsewhere in the API definition; the reference has the following format:  
  
**`#/components/responses/Name`**
    - Description (required): A description of the response. You can use [CommonMark syntax](#) for rich text representation.
  - To reference a response that is defined in a response component, provide the following information on the Reference tab:
    - Response name: A name for the response.  
If you are creating an operation response, this name is the HTTP status code.  
  
If you are creating a response component, this name defines a key that enables this response to be referenced from elsewhere in the API definition; the reference has the following format:  
  
**`#/components/responses/Name`**
    - Reference from: Select a reference to a response component.
2. Click Create.  
The response details are displayed for further editing; see [Editing a response](#).

---

## Editing a response

Responses define the HTTP status code and data returned in a response body and headers. You can edit responses that have been previously created for Path operations in your API definition.

### Before you begin

Open the details form for a response. For details of the areas in your API definition from where you can edit a response, see the following topics:

- [Defining responses for an operation](#)
- [Defining response components](#)



## About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Provide the following information:

- Response name: A name for the response.  
If you are editing an operation response, this name is the HTTP status code.

If you are editing a response component, this name defines a key that enables this response to be referenced from elsewhere in the API definition; the reference has the following format:

**#/components/responses/Name**

To change the name, click Update, then click Save when done.

- Description: A description of the response. You can use [CommonMark syntax](#) for rich text representation.
- Headers: Headers define information that can be returned in the response.  
To define a new header for the response, click Add, then refer to [Creating a header](#).  
  
To edit an existing header, click the header name, then refer to [Editing a header](#).
- Content: A content definition describes the content of the response.  
To define a new content definition for the response, click Add, then refer to [Creating a content definition](#).  
  
To edit an existing content definition, click the content definition name, then refer to [Editing a content definition](#).
- Links: Links describe how various values returned by one operation can be used as input for other operations.  
To define a new link for the response, click Add, then refer to [Creating a link](#).  
  
To edit an existing link, click the link name, then refer to [Editing a link](#).

2. Click Save when done.

## What to do next

---

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

## Creating a content definition

---

A content definition describes the content of an API request, response, or parameter. You can create content definitions in various places in your API definition.

## Before you begin

---

Launch the content definition creation window. For details of the areas in your API definition from where you can create a content definition, see the following topics:

- [Editing a header](#)
- [Editing a request body](#)
- [Defining the request body for an operation](#)
- [Editing a response](#)



## About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Provide the following information:

- Content Type: Select the content type; for example, application/json.

Note: While the Content Type field in the editor provides a selection list of media types, by editing the OpenAPI source for the API directly you can specify a [media type range](#); for example, `text/*, */json`, or `*/*`. For requests or responses that match multiple types, only the most specific type is applicable; for example, `text/plain` overrides `text/*`.

2. Click Create.

The content definition details are displayed for further editing; see [Editing a content definition](#).

---

## Editing a content definition

A content definition describes the content of an API request, response, or parameter. You can edit content definitions that have been previously created in various places in your API definition.

## Before you begin

---

Open the details form for a content definition. For details of the areas in your API definition from where you can edit a content definition, see the following topics:

- [Editing a header](#)
- [Editing a request body](#)
- [Defining the request body for an operation](#)
- [Editing a response](#)



## About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Provide the following information:

- Content Type: Select the content type; for example, application/json.

Note: While the Content Type field in the editor provides a selection list of media types, by editing the OpenAPI source for the API directly you can specify a [media type range](#); for example, `text/*, */json`, or `*/*`. For requests or responses that match multiple types, only the most specific type is applicable; for example, `text/plain` overrides `text/*`.

- Schema:

To define a schema for the content definition, click Create, then refer to [Creating a schema](#).

If the content definition already has a schema defined, click View to edit the schema.

For full details on editing a schema, see [Editing a schema](#).

Note: You cannot edit the schema if it references a schema component, the schema configuration is inherited from the schema component; for details on configuring a schema component, see [Defining schema components](#). You can, however, edit the referenced schema component but any changes will be reflected anywhere that the schema component is referenced.

- Example Value: An example of a parameter value. Whatever you enter here is displayed as-is in the Developer Portal.

- Examples:

To add an example of the content, click Add, then refer to [Creating an example](#).

To edit an existing example, click the example name, then refer to [Editing an example](#).

- Encoding: An encoding definition describes how a specific schema property is encoded.

To add an encoding definition, click Add, then refer to [Creating an encoding definition](#).

To edit an existing encoding definition, click the name, then refer to [Editing an encoding definition](#).

2. Click Save when done.

## What to do next

---

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

---

## Creating an encoding definition

An encoding definition defines a map between a specific schema property and its encoding information.

## Before you begin

---

Launch the encoding definition creation window when configuring a content definition; see [Editing a content definition](#).



## About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Provide the following information:

- Name: The name of the schema property.
- Content Type: The Content-Type for encoding the property; for example, `application/xml; charset=utf-8, image/png, image/jpeg`.
- Style: Describes how the property value will be serialized depending on the type of the property value. Select one of the following options:
  - form: Form style properties defined by [RFC6570](#).
  - spaceDelimited: Space separated array values.
  - pipeDelimited: Pipe separated array values.
  - deepObject: Provides a simple way of rendering nested objects using form properties.
- Explode: When selected, property values of type `array` or `object` generate separate properties for each value of the array or key-value pair of the map.
- Allow Reserved: Determines whether the property value should allow reserved characters, as defined by [RFC3986](#) (`:/?#[]@!$&'()*+,-=`) to be included without percent-encoding.

2. Click Create.

The encoding definition details are displayed for further editing; see [Editing an encoding definition](#).

## Editing an encoding definition

---

An encoding definition defines a map between a specific schema property and its encoding information

## Before you begin

---

Open the details form for an encoding definition when configuring a content definition; see [Editing a content definition](#).



## About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Provide the following information:

- Name: The name of the schema property. To change the name, click Update, then click Save when done.
- Content Type: The Content-Type for encoding the property; for example:
  - `application/xml; charset=utf-8`
  - `image/png, image/jpeg`.
- Headers: A map allowing additional information to be provided as headers.  
To add a new header, click Add, then refer to [Creating a header](#).  
  
To edit an existing header, click the header name, then refer to [Editing a header](#).
- Style: Describes how the property value will be serialized depending on the type of the property value. Select one of the following options:
  - form: Form style properties defined by [RFC6570](#).
  - spaceDelimited: Space separated array values.
  - pipeDelimited: Pipe separated array values.
  - deepObject: Provides a simple way of rendering nested objects using form properties.
- Explode: When selected, property values of type `array` or `object` generate separate properties for each value of the array or key-value pair of the map.
- Allow Reserved: Determines whether the property value should allow reserved characters, as defined by [RFC3986](#) (`:/?#[]@!$&'()*+,-=`) to be included without percent-encoding.

2. Click Save when done.

## What to do next

---

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

## Defining parameter components

Parameter components define reusable parameters that can specify variable elements of a URL path, query parameters, headers, or a request body.

### About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).



You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To make use of a parameter component, you must reference it from elsewhere in your API definition.

A parameter is similar to a header, with the following differences:


- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **header**.
- A parameter can have various format styles, the style of a header is always **simple**.

For more information on headers, see [Creating a header](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
  2. Expand Components then, if there are already one or more parameter components defined, expand Parameters.
  3. You can create a new parameter component, or edit any existing parameter component.
    - To create a parameter component, click the add icon  alongside Parameters, then refer to [Creating a parameter](#).
    - To edit an existing parameter component, click the parameter component name in the navigation pane, then refer to [Editing a parameter](#).
- [Creating a parameter](#)  
Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can create parameters for Paths and Path operations in your API definition.
  - [Editing a parameter](#)  
Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can edit parameters that have been previously created for Paths and Path operations in your API definition.

## Creating a parameter

Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can create parameters for Paths and Path operations in your API definition.

### Before you begin

---

Launch the parameter creation window. For details of the areas in your API definition from where you can create a parameter, see the following topics:

- [Defining parameters for a Path](#)
- [Defining parameters for an operation](#)
- [Defining parameter components](#)

### About this task

---

Note:



- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A parameter is similar to a header, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **header**.
- A parameter can have various format styles, the style of a header is always **simple**.

For more information on headers, see [Creating a header](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Create the parameter; you can either create a parameter from scratch, or you can reference a parameter that is defined in a parameter component.

- To create the parameter from scratch, provide the following information on the Definition tab:
  - Name: Available if you are creating a parameter component, this name defines a key that enables this parameter to be referenced from elsewhere in the API definition; the reference has the following format:

```
#/components/parameters/Name
```

- Parameter name (required): The name of the parameter.
- Located In (required): The location of the parameter. Select one of the following options:
  - query: Parameters that are appended to the URL. For example, `url_path?myparam=myvalue`.
  - header: Custom headers that are expected as part of the request.
  - path: The parameter value is part of the operation's URL, enclosed in {}.
  - cookie: Used to pass a specific cookie value to the API.
- Description: A description of the parameter. You can use [CommonMark syntax](#) for rich text representation.
- Style: Describes how the parameter value will be serialized depending on the type of the parameter value. Select one of the following options:
  - matrix: Path-style parameters defined by [RFC6570](#).
  - label: Label style parameters defined by [RFC6570](#).
  - simple: Simple style parameters defined by [RFC6570](#).
  - form: Form style parameters defined by [RFC6570](#).
  - spaceDelimited: Space separated array values.
  - pipeDelimited: Pipe separated array values.
  - deepObject: Provides a simple way of rendering nested objects using form parameters.
- Select the following options as required:
  - Required: Determines whether this parameter is mandatory. If the Located In property is set to path, this option must be selected.
  - Deprecated: Specifies that a parameter is deprecated and should be transitioned out of usage.
  - Allows Empty Value: Allows sending a parameter with an empty value. This is valid only if the Located In property is set to query.
  - Explode: When selected, parameter values of type **array** or **object** generate separate parameters for each value of the array or key-value pair of the map.
  - Allow Reserved: Determines whether the parameter value should allow reserved characters, as defined by [RFC3986](#) (`:/?#[ ]@!$&'()*+,-;=`) to be included without percent-encoding. This property applies only if the Located In property is set to query.
- To reference a parameter that is defined in a parameter component, provide the following information on the Reference tab:
  - Name: Available if you are creating a parameter component, this name defines a key that enables this parameter to be referenced from elsewhere in the API definition; the reference has the following format:

```
#/components/parameters/Name
```

- Parameter name (required, available if you are creating a path or operation parameter): The name of the parameter.
- Reference from: Select a reference to a parameter component.

2. Click Create.

The parameter details are displayed for further editing; see [Editing a parameter](#).

## Editing a parameter

Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can edit parameters that have been previously created for Paths and Path operations in your API definition.

## Before you begin

Open the details form for a parameter. For details of the areas in your API definition from where you can edit a parameter, see the following topics:

- [Defining parameters for a Path](#)
- [Defining parameters for an operation](#)
- [Defining parameter components](#)

## About this task

Note:



- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A parameter is similar to a header, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **header**.
- A parameter can have various format styles, the style of a header is always **simple**.

For more information on headers, see [Creating a header](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Provide the following information:

- Name: Available if you are editing a parameter component, this name defines a key that enables this parameter to be referenced from elsewhere in the API definition; the reference has the following format:

`#/components/parameters/Name`

To change the name, click Update, then click Save when done.

- Parameter name (required): The name of the parameter.
- Located In (required): The location of the parameter. Select one of the following options:
  - query: Parameters that are appended to the URL. For example, `url_path?myparam=myvalue`.
  - header: Custom headers that are expected as part of the request.
  - path: The parameter value is part of the operation's URL, enclosed in {}.
  - cookie: Used to pass a specific cookie value to the API.
- Description: A description of the parameter. You can use [CommonMark syntax](#) for rich text representation.
- Style: Describes how the parameter value will be serialized depending on the type of the parameter value. Select one of the following options:
  - matrix: Path-style parameters defined by [RFC6570](#).
  - label: Label style parameters defined by [RFC6570](#).
  - simple: Simple style parameters defined by [RFC6570](#).
  - form: Form style parameters defined by [RFC6570](#).
  - spaceDelimited: Space separated array values.
  - pipeDelimited: Pipe separated array values.
  - deepObject: Provides a simple way of rendering nested objects using form parameters.
- Select the following options as required:
  - Required: Determines whether this parameter is mandatory. If the Located In property is set to path, this option must be selected.
  - Deprecated: Specifies that a parameter is deprecated and should be transitioned out of usage.
  - Allows Empty Value: Allows sending a parameter with an empty value. This is valid only if the Located In property is set to query.
  - Explode: When selected, parameter values of type `array` or `object` generate separate parameters for each value of the array or key-value pair of the map.
  - Allow Reserved: Determines whether the parameter value should allow reserved characters, as defined by [RFC3986](#) (`:/?#[ ]@!$%&'()*+,-;=`) to be included without percent-encoding. This property applies only if the Located In property is set to query.
- Schema:
  - To define a schema for the parameter, click Create, then refer to [Creating a schema](#).

If the parameter already has a schema defined, click View to edit the schema.

For full details on editing a schema, see [Editing a schema](#).

Note: You cannot edit the schema if it references a schema component, the schema configuration is inherited from the schema component; for details on configuring a schema component, see [Defining schema components](#). You can, however, edit the referenced schema component but any changes will be reflected anywhere that the schema component is referenced.

- Example Value: An example of a parameter value. Whatever you enter here is displayed as-is in the Developer Portal.

2. Click Save when done.

## What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

## Defining example components

An example component is a reusable example that you can add to an API to make the specification of your API clearer.



## About this task

Note:


- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To make use of an example component, you must reference it from elsewhere in your API definition.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Components then, if there are already one or more example components defined, expand Examples.
3. You can create a new example component, or edit any existing example component.
  - To create an example component, click the add icon  alongside Examples, then refer to [Creating an example](#).
  - To edit an existing example component, click the example component name in the navigation pane, then refer to [Editing an example](#).

- [Creating an example](#)  
You add examples to an API to make the specification of your API clearer.
- [Editing an example](#)  
You add examples to an API to make the specification of your API clearer.

---

## Creating an example

You add examples to an API to make the specification of your API clearer.

---

### Before you begin

Launch the example creation window. For details of the areas in your API definition from where you can create an example, see the following topics:

- [Editing a content definition](#)
- [Defining example components](#)



---

### About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

### Procedure

1. Create the example; you can either create an example from scratch, or you can reference an example that is defined in an example component.
  - To create the example from scratch, provide the following information on the Definition tab:
    - Example Name: If you are creating an example component, this name defines a key that enables this example to be referenced from elsewhere in the API definition; the reference has the following format:  

```
#/components/examples/Name
```
    - Summary: A summary description of the example.
    - Description: A full description of the example. You can use [CommonMark syntax](#) for rich text representation.
    - External Value: A URL that points to a literal example. This provides the capability to reference examples that cannot easily be included in JSON or YAML documents. This could be a PDF, for example.
  - To reference an example that is defined in an example component, provide the following information on the Reference tab:
    - Example Name: If you are creating an example component, this name defines a key that enables this example to be referenced from elsewhere in the API definition; the reference has the following format:  

```
#/components/examples/Name
```
    - Reference from: Select a reference to an example component.
2. Click Create.  
The example details are displayed for further editing; see [Editing an example](#).

---

## Editing an example

You add examples to an API to make the specification of your API clearer.

---

### Before you begin

Open the details form for an example. For details of the areas in your API definition from where you can edit an example, see the following topics:

- [Editing a content definition](#)
- [Defining example components](#)

---



### About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.



At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Provide the following information:

- Example Name: If you are editing an example component, this name defines a key that enables this example to be referenced from elsewhere in the API definition; the reference has the following format:

**`#/components/examples/Name`**

To change the name, click Update, then click Save when done.

- Summary: A summary description of the example.
- Description: A full description of the example. You can use [CommonMark syntax](#) for rich text representation.
- Example Value: A literal example.
- External Value: A URL that points to a literal example. This provides the capability to reference examples that cannot easily be included in JSON or YAML documents. This could be a PDF, for example.

2. Click Save when done.

## What to do next

---

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

## Defining request body components

---

A request body component is a reusable definition of the structure of the body of an API request.



### About this task

---

Note:


- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
  2. Expand Components then, if there are already one or more request body components defined, expand Request Bodies.
  3. You can create a new request body component, or edit any existing request body component.
    - To create a request body component, click the add icon  alongside Request Bodies, then refer to [Creating a request body](#).
    - To edit an existing request body component, click the request body component name in the navigation pane, then refer to [Editing a request body](#).
- [Creating a request body](#)  
A request body defines the structure of the body of an API request.
  - [Editing a request body](#)  
A request body defines the structure of the body of an API request.

## Creating a request body

---

A request body defines the structure of the body of an API request.

### Before you begin

---

Launch the request body creation window; see [Defining request body components](#).



### About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Create the request body; you can either create a request body from scratch, or you can reference a request body that is defined in a request body component.
  - To create the request body from scratch, provide the following information on the Definition tab:
    - Name: A name for the request body.  
If you are creating a request body component, this name defines a key that enables this request body to be referenced from elsewhere in the API definition; the reference has the following format:  

```
#/components/requestBodies/Name
```
    - Description: A description of the request body. You can use [CommonMark syntax](#) for rich text representation.
    - Required: Determines if the request body is required in the request.
  - To reference a request body that is defined in a request body component, provide the following information on the Reference tab:
    - Name: A name for the request body.  
If you are creating a request body component, this name defines a key that enables this request body to be referenced from elsewhere in the API definition; the reference has the following format:  

```
#/components/requestBodies/Name
```
    - Reference from: Select a reference to a request body component.
2. Click Create.  
The request body details are displayed for further editing; see [Editing a request body](#).

---

## Editing a request body

A request body defines the structure of the body of an API request.

## Before you begin

---

Open the details form for a request body; see [Defining request body components](#).



## About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Provide the following information:
  - Name: A name for the request body.  
Available if you are creating a request body component, this name defines a key that enables this request body to be referenced from elsewhere in the API definition; the reference has the following format:  

```
#/components/requestBodies/Name
```
  - Description: A description of the request body. You can use [CommonMark syntax](#) for rich text representation.
  - Content: A content definition describes the content of the request body.  
To define a new content definition for the request body, click Add, then refer to [Creating a content definition](#).  
  
To edit an existing content definition, click the content definition name, then refer to [Editing a content definition](#).
  - Required: Determines if the request body is required in the request.
2. Click Save when done.

## What to do next

---

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

---

## Defining header components

A header component defines a reusable custom header that can be sent in an API request or returned in an API response.

## About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).



You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To make use of a header component, you must reference it from elsewhere in your API definition.

A header is similar to a parameter, with the following differences:


- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **header**.
- A parameter can have various format styles, the style of a header is always **simple**.

For more information on parameters, see [Creating a parameter](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
  2. Expand Components then, if there are already one or more header components defined, expand Headers.
  3. You can create a new header component, or edit any existing header component.
    - To create a header component, click the add icon  alongside Headers, then refer to [Creating a header](#).
    - To edit an existing header component, click the header component name in the navigation pane, then refer to [Editing a header](#).
- [Creating a header](#)  
A header defines information that can be sent in an API request or returned in an API response. You can create headers in various places in your API definition.
  - [Editing a header](#)  
A header defines information that can be sent in an API request or returned in an API response. You can edit headers that have been previously created in various places in your API definition.

## Creating a header

---

A header defines information that can be sent in an API request or returned in an API response. You can create headers in various places in your API definition.

## Before you begin

---

Launch the header creation window. For details of the areas in your API definition from where you can create a header, see the following topics:

- [Editing a response](#)
- [Editing an encoding definition](#)
- [Defining header components](#)

## About this task

---

Note:


- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A header is similar to a parameter, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **header**.
- A parameter can have various format styles, the style of a header is always **simple**.

For more information on parameters, see [Creating a parameter](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Create the header; you can either create a header from scratch, or you can reference a header that is defined in a header component.
  - To create the header from scratch, provide the following information on the Definition tab:
    - Header name: If you are creating a header component, this name defines a key that enables this header to be referenced from elsewhere in the API definition; the reference has the following format:  
**`#/components/headers/Name`**
    - Description: A description of the header. You can use [CommonMark syntax](#) for rich text representation.
    - Required: Determines whether this header is mandatory.
    - Deprecated: Specifies that this header is deprecated and should be transitioned out of usage.

- Allow Empty Value: The header can be sent with an empty value.
- Explode: When selected, header values of type **array** or **object** generate separate headers for each value of the array or key-value pair of the map.
- Allow Reserved: Determines whether the header value should allow reserved characters, as defined by [RFC3986](#) (:/?#[ ]@!\$&'()\*+,-;=) to be included without percent-encoding.
- To reference a header that is defined in a header component, provide the following information on the Reference tab.
  - Header name: If you are creating a header component, this name defines a key that enables this header to be referenced from elsewhere in the API definition; the reference has the following format:

**#/components/headers/Name**

- Reference from: Select a reference to a header component.

2. Click Create.

The header details are displayed for further editing; see [Editing a header](#).

---

## Editing a header

A header defines information that can be sent in an API request or returned in an API response. You can edit headers that have been previously created in various places in your API definition.

---

## Before you begin

Open the details form for a header. For details of the areas in your API definition from where you can edit a header, see the following topics:

- [Editing a response](#)
- [Editing an encoding definition](#)
- [Defining header components](#)

---

## About this task

Note:



- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A header is similar to a parameter, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **header**.
- A parameter can have various format styles, the style of a header is always **simple**.

For more information on parameters, see [Creating a parameter](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

## Procedure

1. Provide the following information:

- Header name: If you are editing a header component, this name defines a key that enables this header to be referenced from elsewhere in the API definition; the reference has the following format:

**#/components/headers/Name**

To change the name, click Update, then click Save when done.

- Description: A description of the header. You can use [CommonMark syntax](#) for rich text representation.
- Required: Determines whether this header is mandatory.
- Deprecated: Specifies that this header is deprecated and should be transitioned out of usage.
- Allow Empty Value: The header can be sent with an empty value.
- Explode: When selected, header values of type **array** or **object** generate separate headers for each value of the array or key-value pair of the map.
- Allow Reserved: Determines whether the header value should allow reserved characters, as defined by [RFC3986](#) (:/?#[ ]@!\$&'()\*+,-;=) to be included without percent-encoding.
- Schema:
  - To define a schema for the header, click Create, then refer to [Creating a schema](#).

If the header already has a schema defined, click View to edit the schema.

For full details on editing a schema, see [Editing a schema](#).

Note: You cannot edit the schema if it references a schema component, the schema configuration is inherited from the schema component; for details on configuring a schema component, see [Defining schema components](#). You can, however, edit the referenced schema component but any changes will be reflected anywhere that the schema component is referenced.

- Content: A content definition describes the content of the header.
  - To define a new content definition for the header, click Add, then refer to [Creating a content definition](#).

To edit an existing content definition, click the content definition name, then refer to [Editing a content definition](#).

2. Click Save when done.

## What to do next

---

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

## Defining security scheme components

A security scheme component specifies all the settings for a particular aspect of API security; for example, the user registry that you use to authenticate access to the API.

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can create security definitions of the following types:

Type	Description
Basic authentication	Use a basic authentication security definition to specify a user registry or an authentication URL to be used to authenticate access to the API.
API key	Use an API key security definition to specify what application credentials are required to call an API.
OAuth2	Use an OAuth2 security definition to specify settings for OAuth token based authentication for your API.
HTTP Bearer	Use an HTTP Bearer definition to specify how to validate the Bearer token that is required to call an API.

- [Defining basic authentication security scheme components](#)  
A basic authentication security scheme component is used when an application that calls the API is required to authenticate through a user registry.
- [Defining API key security scheme components](#)  
An API key security component is used to specify the credentials that an application must provide to identify itself when calling the API operations.
- [Defining OAuth2 security scheme components](#)  
An OAuth2 security scheme component defines the settings for controlling access to the API operations through the OAuth authorization standard.
- [Defining an HTTP bearer security scheme](#)  
An HTTP bearer security scheme is used to generate access tokens that are exchanged between the server and the client when calling the API operations.

## Defining basic authentication security scheme components

A basic authentication security scheme component is used when an application that calls the API is required to authenticate through a user registry.

### Before you begin

---

For authenticating access to the operations of APIs that are enforced on the API Connect gateway, the following user registry types are supported:

- Authentication URL
- LDAP

The following user registry types are **not** supported:

- OpenID Connect (OIDC)
- Local User Registry (LUR)
- A custom user registry

Before you can create a basic authentication security definition in an API, the user registry must exist. To create a user registry, you can use either API Manager or Cloud Manager. When you create a registry in API Manager, it is visible only to your provider organization. When you create a registry in Cloud Manager, you can make it visible to multiple provider organizations.

To create a user registry with API Manager, see [Working with user registries](#).

To create a user registry with Cloud Manager, see [User registries overview](#).

### About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.



When you use basic authentication, you require API users to provide a valid user name and password to access selected operations. The application developer must also provide an HTTP authorization header in requests that are sent to operations that require basic authentication.

When you use an authentication URL, the user credentials that are provided in the authorization header are validated by the endpoint specified in the URL. If the user is authenticated, IBM API Connect expects an authentication URL to return an HTTP 200 OK response status code. All other HTTP response status codes result in an authentication failure and access is denied.

For more information about using an LDAP user registry for authentication, see [LDAP authentication](#).


For information about using an Authentication URL, see [Authentication URL user registry](#).

To make use of a basic authentication security scheme component, you must reference it from elsewhere in your API definition. For more information, see [Enforcing security requirements on an API](#) and [Enforcing security requirements on an operation](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Components then, if there are already one or more security scheme components defined, expand Security Schemes.
3. To create a basic authentication security scheme component, click the add icon  alongside Security Schemes. To edit an existing basic authentication security scheme component, click the component name in the navigation pane.
4. If you are creating a new basic authentication security scheme component, you can either create it from scratch or you can reference another basic authentication security scheme component.

To create a basic authentication security scheme component from scratch, provide the following information on the Definition tab:

- Security Scheme Name (required): A key that enables this basic authentication security scheme component to be referenced from elsewhere in the API definition; the reference has the following format:

```
#/components/securitySchemes/Security_Scheme_Name
```

- Security Scheme Type (required): Select http.
- Scheme (required): The name of the HTTP Authorization scheme to be used in the Authorization header as defined in [RFC7235](#).
- Authenticate using User Registry: The name of the user registry to be used for authentication. This is the value of the `name` field that was auto-generated when the user registry was created.
- Bearer format: A hint to the client to identify how the bearer token is formatted. Bearer tokens are usually generated by an authorization server, so this information is primarily for documentation purposes.
- Description: An optional description of the basic authentication security scheme. You can use [CommonMark syntax](#) for rich text representation.

To reference another basic authentication security scheme component, provide the following information on the Reference tab:

- Security Scheme Name (required): A key that enables this basic authentication security scheme component to be referenced from elsewhere in the API definition; the reference has the following format:

```
#/components/securitySchemes/Security_Scheme_Name
```

- Reference from: Select a reference to another basic authentication security scheme component.
5. If you are creating a new basic authentication security scheme component, click Create.  
The security scheme component details are displayed for further editing.
  6. Click Save when done.

## What to do next

---

Apply the security scheme to an API or operation. For more information, see [Enforcing security requirements on an API](#) and [Enforcing security requirements on an operation](#).

For more information on LDAP and Authentication URL, see [LDAP authentication](#) and [Authentication URL user registry](#).

## Defining API key security scheme components

---

An API key security component is used to specify the credentials that an application must provide to identify itself when calling the API operations.

## About this task

---



Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can require that, when calling an API operation, an application must provide either a client ID, or a client ID and client secret; you create an API key security scheme component to specify a credentials requirement. If you require that an application must provide both a client ID and client secret, you must create two API key security scheme components, one for each type of credentials.


To make use of an API key security scheme component, you must reference it from elsewhere in your API definition. For more information, see [Enforcing security requirements on an API](#) and [Enforcing security requirements on an operation](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Components then, if there are already one or more security scheme components defined, expand Security Schemes.

- To create an API key security scheme component, click the add icon  alongside Security Schemes. To edit an API key existing security scheme component, click the component name in the navigation pane.
- If you are creating a new API key security scheme component, you can either create it from scratch or you can reference another API key security scheme component.

To create an API key security scheme component from scratch, provide the following information on the Definition tab:

- Security Scheme Name: A key that enables this API key security scheme component to be referenced from elsewhere in the API definition; the reference has the following format:

```
#/components/securitySchemes/Security_Scheme_Name
```

- Security Scheme Type: Select apiKey.
- Name: The name of the header, query or cookie parameter to be used. For a key of type client secret you must include the string "secret", ignoring case, in the value you specify in the Name field; if the value does not contain the string "secret" then the key is assumed to be of type client ID.  
Note: The client ID and client secret headers that are specified in the request when the API is called are **not** added automatically to the message context. If you need these headers in the message context for subsequent processing, include a **set-variable** policy in your API assembly that adds the headers to the message content, taking the values specified in the request; you can do this in either of the following ways:

- In the Assemble tab, add a Set Variable policy to your API assembly that defines the appropriate actions; for example:

Table 1.

Action	Set	Type	Value
Set	message.headers.X-IBM-Client-Id	string	\$(request.headers.X-IBM-Client-Id)
Set	message.headers.X-IBM-Client-Secret	string	\$(request.headers.X-IBM-Client-Secret)

For more information on configuring a Set Variable policy, see [Set Variable](#).

- In the Source tab, add a **set-variable** policy directly to the assembly section in your OpenAPI source; for example:

```
assembly:
 execute:
 - set-variable:
 version: 2.0.0
 title: set-variable
 actions:
 - set: message.headers.X-IBM-Client-Id
 value: $(request.headers.X-IBM-Client-Id)
 type: string
 - set: message.headers.X-IBM-Client-Secret
 value: $(request.headers.X-IBM-Client-Secret)
 type: string
```

For more information on defining a **set-variable** policy in your OpenAPI source, see [set-variable](#).

You should position the **set-variable** before any policy that needs to access the API key headers; for example, an invoke policy that calls a back-end service that is required to identify the application that made the initial request.

- Key Type: Select client\_id or client\_secret.

Important: The client\_id is not confidential and functions as a "SecurityScheme" only in the meaning of the OpenAPI specification.

If you use only the client\_id (without the client\_secret) as your API Key, be aware that the client\_id is not considered a confidential value in the API Manager UI; for example, it might be displayed to other authenticated members of the provider organization. The client\_id is treated as a public identifier for apps and cannot be kept private.

If authentication of the identity of the app calling an API is needed, then the API Key should use both a client\_id to identify the app, plus a client\_secret to authenticate it. The client\_secret is designed to be known only to the app and the authorization server, and it should be protected. IBM API Connect protects the client\_secret: it is only available when first generated, it can be stored as a one-way hash to prevent retrieval or leaks, and it can be changed and reset without changing the corresponding client\_id. When an API Key is not marked as a client\_secret, it will not be protected in the API Manager UI.

- Located In: Select one of the following options, to specify how the credentials are sent:
  - header: The credentials are sent in the request header.
  - query: The credentials are sent as query parameters. This method is less secure because the client secret could be exposed in a log file.

Note: If you migrated from API Connect V5, API key security definitions that contain multiple Client ID references, behave differently in Version 10 than in Version 5. For information about the differences, see [Client ID behavior in API key security definitions](#).

- Description: An optional description of the API key security scheme. You can use [CommonMark syntax](#) for rich text representation.

- To reference another API key security scheme component, provide the following information on the Reference tab:

- Security Scheme Name: A key that enables this API key security scheme component to be referenced from elsewhere in the API definition; the reference has the following format:

```
#/components/securitySchemes/Security_Scheme_Name
```

- Reference from: Select a reference to another API key security scheme component.

- If you are creating a new API key security scheme component, click Create.

The security scheme component details are displayed for further editing.

- Click Save when done.

## What to do next

Apply the security scheme to an API or operation. For more information, see [Enforcing security requirements on an API](#) and [Enforcing security requirements on an operation](#).

- [Client ID behavior in API key security definitions](#)

How the Client ID behaves in API key security definitions in IBM API Connect 10 compared to Version 5.

## Client ID behavior in API key security definitions

How the Client ID behaves in API key security definitions in IBM® API Connect 10 compared to Version 5.

API key security definitions that contain multiple Client ID references, behave differently in Version 10 than in Version 5. Refer to the following table to understand the differences in Client ID behavior for different use cases.

Note: Although a client\_id in the request body is supported, this data is not used during security requirement processing.

Table 1. Client ID behavior in V5 and V10

Use case	Description of security requirement	V5 behavior	V10 behavior
1	Allows client_id in the query, and the request contains multiple client_ids in the query.	Uses the first client ID in the request.	401 invalid client ID or secret.
2	Allows client_id in the header, and the request contains multiple client_ids in the header.	401 invalid client ID or secret.	401 invalid client ID or secret.
3	Allows client_id in the header, and the request contains multiple client_ids in the query and the header (1 in the query, 1 in the header).	401 client ID in wrong location.	Client ID in query is ignored.
4	Allows client_id in the query, and the request contains multiple client_ids in the query and the header.	Client ID in header is ignored.	Client ID in header is ignored.
5	Allows client_id in the query or header, and both the query and header contain multiple client_ids.	Header returns 401 client ID in wrong location. Query uses the first client ID in the request.	401 invalid client ID or secret.
6	Allows client_id in the header, and the request contains a valid client_id in the query, and an invalid client_id in the header.	401 client ID in wrong location.	401 invalid client ID or secret.
7	Allows client_id in the query, and the request contains a valid client_id in the header, and an invalid client_id in the query.	401 invalid client ID or secret.	401 invalid client ID or secret.
8	Allows client_id in the query or header, and both the query and header contain valid client_ids.	200 OK.	403 multiple client IDs.
9	Allows client_id in the query or header, and both the query and header contain invalid client_ids.	401 client ID in wrong location.	401 invalid client ID or secret.
10	Allows client_id in the query or header, and both the query and header contain a client_id, and the one in the query is valid.	200 OK.	Uses the valid one 200 OK.
11	Allows client_id in the query or header, and both the query and header contain a client_id, and the one in the header is valid.	401 client ID in wrong location.	Uses the valid one 200 OK.
12	Allows client_id in the query or header, and both the query and client_id is missing in the request (no security requirement, but client security is defined in the assembly).	401 invalid client ID or secret.	<ul style="list-style-type: none"> <li>When Return V5 Responses compatibility option is off: 401 client ID is missing.</li> <li>When Return V5 Responses compatibility option is on: 401 invalid client ID or secret.</li> </ul> <p>For information about setting compatibility options, see <a href="#">Configuring v5 compatibility options</a>.</p>

## Defining OAuth2 security scheme components

An OAuth2 security scheme component defines the settings for controlling access to the API operations through the OAuth authorization standard.

### Before you begin

Before you can create an OAuth2 security scheme component, you must:

1. Create an OAuth provider.
  - To use Cloud Manager, see [Configuring a native OAuth provider](#) or [Configuring a third-party OAuth provider](#).
  - To use API Manager, see [Configuring a native OAuth provider](#) or [Configuring a third-party OAuth provider](#).
2. Add the OAuth provider to a catalog. If you have not created any catalogs, use the Sandbox Catalog. See the [OAuth instructions step](#) in [Creating and configuring Catalogs](#).


### About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.


To make use of an OAuth2 security scheme component, you must reference it from elsewhere in your API definition. For more information, see [Enforcing security requirements on an API](#) and [Enforcing security requirements on an operation](#).

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Components then, if there are already one or more security scheme components defined, expand Security Schemes.



3. To create an OAuth2 security scheme component, click the add icon  alongside Security Schemes. To edit an existing OAuth2 security scheme component, click the component name in the navigation pane.
4. If you are creating a new OAuth2 security scheme component, you can either create it from scratch or you can reference another OAuth2 security scheme component.

To create an OAuth2 security scheme component from scratch, provide the following information on the Definition tab:

- Security Scheme Name: A key that enables this OAuth2 security scheme component to be referenced from elsewhere in the API definition; the reference has the following format:

```
#/components/securitySchemes/Security_Scheme_Name
```

- Security Scheme Type: Select oauth2.
- Flow Type: Select one or more of the following options:
  - implicit
  - password
  - clientCredentials
  - authorizationCode
- OAuth Provider: The name of the OAuth provider to be used. This is the value of the **name** field that was auto-generated when the OAuth provider was created.
- Description: An optional description of the OAuth2 security scheme. You can use [CommonMark syntax](#) for rich text representation.

To reference another OAuth2 security scheme component, provide the following information on the Reference tab:

- Security Scheme Name: A key that enables this OAuth2 security scheme component to be referenced from elsewhere in the API definition; the reference has the following format:


```
#/components/securitySchemes/Security_Scheme_Name
```

- Reference from: Select a reference to another OAuth2 security scheme component.

5. If you are editing an existing OAuth2 security scheme component, provide the following information:

- The URL endpoints (the applicable endpoints depend on the flow type):
  - Authorization URL
  - Refresh URL
  - Token URL

Note: The URL endpoint values supplied here are maintained only for informative purposes, no validation or other action is applied to them by API Connect. The OAuth security that is applied depends on the specified OAuth provider.

- Scopes:
  - To add a new scope, click Add. To edit an existing scope, click the options icon  alongside the required scope, then click Edit.
  - Provide the following information:
    - Scope: The scope identifier.
    - Description: An optional description of the scope. You can use [CommonMark syntax](#) for rich text representation.

6. If you are creating a new OAuth2 security scheme component, click Create.

The security scheme component details are displayed for further editing.

7. Click Save when done.

## What to do next

Apply the security scheme to an API or operation. For more information, see [Enforcing security requirements on an API](#) and [Enforcing security requirements on an operation](#).

## Defining an HTTP bearer security scheme

An HTTP bearer security scheme is used to generate access tokens that are exchanged between the server and the client when calling the API operations.

## About this task



For information on the use of bearer tokens with OpenAPI 3 APIs, see [Bearer Authentication](#) in the OpenAPI 3 specification.

If an HTTP bearer token is found in the request, its value is stored within the context as `api->security->bearer_token`. If an external URL validation is invoked, any response with a status code of 200 is stored within the context as `api->security->bearer_validation_response`.


Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Expand Components, , Security Schemes.
3. Click Add.
4. In the Add Object dialog box, provide the following information:
  - Security Scheme Name (Key) - Provide a descriptive name for the new scheme.

- Security Scheme Type - Select http.
  - Scheme - Select Bearer.
  - Bearer Format - Select JWT.
  - Validation Method - Select a method. If you select external-url, provide the following additional information:
    - Validation Endpoint - Provide the URL of the server used for validation. To ensure a secure connect, the URL should use the **https** protocol.
    - TLS Profile name - If the validation endpoint uses the **https** protocol, select the name of the TLS client profile to use for a secure connection.
  - Description - Provide a description of the JWT scheme.
5. Click Create to create the new scheme.
  6. Click Save in the page header.

---

## Defining link components

A link component is a reusable link definition that can be used to describe how various values returned by one operation can be used as input for other operations. In this way, links provide a known relationship and traversal mechanism between the operations.

---



### About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).


You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To make use of a link component, you must reference it from elsewhere in your API definition.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

### Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
  2. Expand Components then, if there are already one or more link components defined, expand Links.
  3. You can create a new link component, or edit any existing link component.
    - To create a link component, click the add icon  alongside Links, then refer to [Creating a link](#).
    - To edit an existing link component, click the link component name in the navigation pane, then refer to [Editing a link](#).
- [Creating a link](#)  
Links describe how various values returned by one operation can be used as input for other operations, providing a known relationship and traversal mechanism between the operations.
  - [Editing a link](#)  
Links describe how various values returned by one operation can be used as input for other operations, providing a known relationship and traversal mechanism between the operations.

---

## Creating a link

Links describe how various values returned by one operation can be used as input for other operations, providing a known relationship and traversal mechanism between the operations.

---

### Before you begin

Launch the link creation window. For details of the areas in your API definition from where you can create a link, see the following topics:

- [Editing a response](#)
- [Defining link components](#)



---

### About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

---

### Procedure

1. Create the link; you can either create a link from scratch, or you can reference a link that is defined in a link component.

- To create the link from scratch, provide the following information on the Definition tab:
  - Name: If you are creating a link component, this name defines a key that enables this link to be referenced from elsewhere in the API definition; the reference has the following format:
 

```
#/components/links/Name
```
  - Operation ID: The ID of an existing operation; this field is mutually exclusive of the Operation Ref field. For details on defining operations, see [Defining operations for a Path](#).
  - Operation Ref: A relative or absolute URI reference to an existing operation; this field is mutually exclusive of the Operation ID field. For details on defining operations, see [Defining operations for a Path](#).
  - Description: A description of the link. You can use [CommonMark syntax](#) for rich text representation.
- To reference a link that is defined in a link component, provide the following information on the Reference tab:
  - Name: If you are creating a link component, this name defines a key that enables this link to be referenced from elsewhere in the API definition; the reference has the following format:
 

```
#/components/links/Name
```
  - Reference from: Select a reference to a link component.

2. Click Create.

The link details are displayed for further editing; see [Editing a link](#).

## Editing a link

Links describe how various values returned by one operation can be used as input for other operations, providing a known relationship and traversal mechanism between the operations.

## Before you begin

Open the details form for a link. For details of the areas in your API definition from where you can edit a link, see the following topics:



- [Editing a response](#)
- [Defining link components](#)

## About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Provide the following information:

- Name: If you are editing a link component, this name defines a key that enables this link to be referenced from elsewhere in the API definition; the reference has the following format:

```
#/components/links/Name
```

To change the name, click Update, then click Save when done.

- Operation ID: The ID of an existing operation; this field is mutually exclusive of the Operation Ref field. For details on defining operations, see [Defining operations for a Path](#).
- Operation Ref: A relative or absolute URI reference to an existing operation; this field is mutually exclusive of the Operation ID field. For details on defining operations, see [Defining operations for a Path](#).
- Parameters: A map representing parameters to pass to an operation as specified with Operation ID or identified via Operation Ref. The key is the parameter name to be used, whereas the value can be a constant or an expression to be evaluated and passed to the linked operation.
- Request Body: The request body to use when calling the target operation.
- Description: A description of the link. You can use [CommonMark syntax](#) for rich text representation.
- Server: A server to be used by the target operation. Provide the following information:
  - Server URL The specified URL is used to determine the full URL endpoint for the calling the API, according to any vanity endpoint configuration in the Catalog in which the API is published. For full details, see [Configuring vanity endpoints for a Catalog](#).
  - Server Description: An optional description of the host designated by the URL. You can use [CommonMark syntax](#) for rich text representation.
  - Server Variables A server variable defines a map between a variable name and its value. The value is used for substitution in the server's URL template.
    - To add a new server variable, click Add. To edit an existing server variable, click the variable name.
    - Provide the following information:
      - Server Variable Name (required).
      - Default value (required).
      - An optional rich text description. You can use [CommonMark syntax](#) for rich text representation.
      - One or more Enum Value entries (available when editing an existing server variable). Enum values specify an enumeration of string values to be used if the substitution options are from a limited set. To add a new Enum value, click Add, enter the value, then click Create. To edit an existing Enum value, click the Enum value.
    - If you are creating a new server variable, click Create. The server variable details are displayed for further editing.

2. Click Save when done.

## What to do next

---

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

---

## Specifying gateway and portal settings

Define general configuration settings for your API.



### About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Select the Gateway tab, then click Gateway and portal settings.
3. Provide the following information:
  - Phase: The phase of the lifecycle that your API is in; select one of the following options:
    - Realized (default) - The API is in the implementation phase.
    - Identified - The API is in the early conceptual phase and is neither fully designed nor implemented.
    - Specified - The API has been fully designed and passed an internal milestone but has not yet been implemented.
  - Testable: Select this option to allow the API's operations to be tested using the test tool in the Developer Portal.  
Note: For the test tool to work, an API must be included in a Plan in a Product that is staged in a Catalog.
  - CORS: Enable and configure cross-origin resource sharing (CORS) support for your API. For details, see [Enabling CORS support for an API](#).
  - Target Services: Define web services that you want to use in your API definition. For details, see [Defining target services for an API](#).
  - Enforced: Select this option to enforce the API by using the DataPower API Gateway. Clear this option if you are managing the API on a gateway other than the DataPower API Gateway. Although not published to a gateway by API Connect, an unenforced API is still available in the Developer Portal for subscription by application developers.
  - Properties: Add properties that can be referenced in your API definition. For details, see [Setting API properties](#).
  - Catalog Properties: Define property values that are specific to a particular Catalog. For details, see [Defining Catalog specific property values](#).
  - Activity Log: Configure your logging preferences for the API activity that is stored in Analytics. For details, see [Configuring activity logging](#).
  - Extensions: Extensions added to the API to extend the OpenAPI specification. For details, see [Adding an OpenAPI extension to an API](#).

---

## Defining target services for an API

A target service defines a web service that you want to use in your API definition.



### About this task

---

Note:


- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

### Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Select the Gateway tab, expand Gateway and portal settings then, if there are already one or more target services defined, expand Target Services.
3. To define a new API target service, complete the following steps:
  - a. Click the add icon  alongside Target Services in the navigation pane.
  - b. Upload the service information from a stand-alone .wsdl file, or a .zip file that contains a WSDL file and its dependent documents, by either dragging and dropping your file, or browsing and selecting the file that you want to use.  
If you upload a .zip file, you can include in the .zip file an options file to specify additional directives. For details, see [Using an options file when importing a WSDL service](#).

4. Select the service that you want to add.
5. Click Save to save your changes.
6. To view the source for an existing target service, click the target service name in the navigation pane.

## Results

---

The target services that you add are now available in the palette on the Policies view for you to add to your API assembly flow; for more information, see [Including elements in your API assembly](#).

## Setting API properties

---

In addition to the pre-supplied API properties that you can use to control the behavior of API Connect policies, you can define your own API properties. The properties that you define can be referenced in your API definitions.

## About this task

---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).



You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

API properties include property name, value, and, optionally a specific Catalog to which a property value applies. For a list of pre-supplied API properties relating to various policies, see [API properties](#).

Note: Once defined, an API property is read only.

For information on how to reference a property in an API definition, see [Variable references in API Connect](#).


It is also possible to define properties that are specific to a Catalog and can be referenced by any of the APIs in that Catalog; for more information, see [Creating and configuring Catalogs](#). Note that if you define a Catalog property of the same name as an API property, the API property takes precedence over the Catalog property.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Tip: If you add or change an API property on an API that is already staged or published, you must re-stage or re-publish the Product that contains the updated API for the change to take effect.

## Procedure

---

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Select the Gateway tab, expand Gateway and portal settings then, if there are already one or more API properties defined, expand Properties.
3. Configure an API property.
  - To define a new API property, complete the following steps:
    - Click the add icon  alongside Properties in the navigation pane.
    - Provide the following information:
      - Property Name: Enter a name for the property; this name is used to reference the property. The following character set is supported for the Name of an API property: `[A-Za-z0-9_-]+`. Spaces are allowed.
      - Value: A default value for the property. Leave blank if the property is to have a null value by default.
      - Description: An optional description of the property. You can use [CommonMark syntax](#) for rich text representation.
      - Select Encoded if you want to hide the property values, or protect user passwords from casual observance.  
Note: If you encode a property value, it is saved in Base64 encoded form; it is **not** encrypted. If you subsequently clear the Encoded check box, the original property value is restored in its unencoded form.
    - Click Create.
  - To modify an existing API property, click the property name in the navigation pane. You can then change any of the configuration settings.
4. Click Save to save your changes.

## What to do next

---

Optionally, define a value for the property that is specific to a particular Catalog; see [Defining Catalog specific property values](#).

- [Defining Catalog specific property values](#)  
For any API property, you can define a value that is specific to a particular Catalog. For any API published to that Catalog, the API property assumes the Catalog specific value, overriding any default property value.

## Defining Catalog specific property values

---

For any API property, you can define a value that is specific to a particular Catalog. For any API published to that Catalog, the API property assumes the Catalog specific value, overriding any default property value.

## Before you begin

---



You must have previously created the API property that you now want to define a Catalog specific value for; see [Setting API properties](#).

## About this task

Note:


- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Tip: If you add or change an API property on an API that is already staged or published, you must re-stage or re-publish the Product that contains the updated API for the change to take effect.

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Select the Gateway tab, expand Gateway and portal settings then, if there are already one or more Catalog specific property values defined, expand Catalog Properties.
3. Configure a Catalog specific property value.
  - To define a new Catalog specific property value, complete the following steps:
    - Click the add icon  alongside Catalog Properties in the navigation pane.
    - Type the Catalog Name exactly as it displays in the **name** field for the selected Catalog.  
Tip: For the best result, manually type the catalog's name instead of selecting its title. Using the catalog name is safer than using its title, because the title might not be unique within the provider organization and might be modified later (which breaks the association between the property and the catalog). The catalog's name is always unique within the provider organization and cannot be changed.
    - Click Create. The Catalog properties details for the specified catalog are displayed for further editing.
    - Alongside Property Overrides, click Add.
    - Provide the following information:
      - Property Name: Enter the name of the API property that you want to define a Catalog specific value for. The name **must** match the name of a previously defined API property; see [Setting API properties](#).
      - Property Value: The Catalog specific property value. Leave blank if the property is to have a null value by default.
    - Click Create.
  - To modify an existing Catalog specific property value, click the Catalog name in the navigation pane. You can then change the Catalog name, and add, modify, or delete property override settings.
4. Click Save to save your changes.

## Configuring activity logging

You can configure your logging preferences for the API activity that is stored in Analytics, overriding the default activity logging behavior.

## About this task

Note:



- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

An API event record exists for each API execution event in the Gateway server. By default, the content type for a successful API execution is activity, and payload for an API call the results in an error code. When you compose your API definition, you can change the type of content to log in these API event records. During API execution, the activity data is stored in the `log` context variable, which populates the API event record on completion of the API execution; for more information, see [API activity logging context variables](#).

Note:

Activity logging that calls for logging of Analytics data upon success doesn't apply for the OAuth provider. The OAuth provider logs Analytics data for failure cases, but doesn't log successful cases.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

## Procedure

1. Open the API for editing, as described in [Editing an OpenAPI 3.0 API definition](#).
2. Select the Gateway tab, expand Gateway and portal settings, then click Activity Log.
3. Select Enabled, then select options for the Success Content and Error Content fields as follows:

Field label	Description
-------------	-------------

Field label	Description
Success Content	<p>Defines the type of content to be logged when the operation is successful.</p> <ul style="list-style-type: none"> <li>• none: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal.</li> <li>• activity: Logs invocation only; only the resource URI is recorded.</li> <li>• header: Logs activity and header; the HTTP headers, in addition to the resource URI are recorded.</li> <li>• payload: Logs activity, header, and payload; all information, including the payload that is received in a request or returned in a response, is recorded.</li> </ul> <p>The default value is activity.</p>
Error Content	<p>Indicates what content to log when an error occurs.</p> <ul style="list-style-type: none"> <li>• none: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal.</li> <li>• activity: Logs invocation only; only the resource URI is recorded.</li> <li>• header: Logs activity and header; the HTTP headers, in addition to the resource URI are recorded.</li> <li>• payload: Logs activity, header, and payload; all information, including the payload that is received in a request or returned in a response, is recorded.</li> </ul> <p>The default value is payload.</p>

4. Click Save to save your changes.

## Including elements in your API assembly

An assembly is formed of elements that are applied to calls to and responses from operations in your API. Elements can be either policies or logic constructs.

### About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

In the Policies view of IBM API Connect, you can add and configure elements in your assembly. You can also directly add elements to the OpenAPI definition of your API.

- [The assembly editor](#)  
The API Designer of IBM API Connect features a graphical editor that you can use to create assemblies. With assemblies, you can readily tailor your APIs to include elements such as activity logging and redaction of specific fields.
- [Adding elements to your assembly](#)  
Create your API assembly by using the Assemble view in the API Designer.
- [Handling errors in the assembly](#)  
Use the catch section of the assembly to describe the handling of errors thrown during the assembly execution.

## The assembly editor


The API Designer of IBM® API Connect features a graphical editor that you can use to create assemblies. With assemblies, you can readily tailor your APIs to include elements such as activity logging and redaction of specific fields.

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM API Connect](#).

You can access the Assemble view by clicking the Gateway tab in the API editor, and then click Policies. This view includes a palette, which lists the available elements, a property sheet, which is used to configure an element, and a canvas, which is used to arrange and visualize the assembly's elements.

### The palette

The palette is a list of different elements that you can include in your assembly. The palette can be hidden by clicking the Show/hide policy palette icon .


### The canvas



You can use the canvas to create a graphical representation of the assembly flow. You can drag various elements from the palette to the appropriate location on the canvas. When you drag an element, valid positions are shown by dashed boxes.

Note: You must click Save in the Assemble view to save your assembly updates, and to make them appear in the Source view.

API calls are made at the initial, unfilled, circle, and returned at the final, filled, circle. You can insert elements between the two circles to modify the data received from the call, or returned by the response. To add an element, select it in the palette and drag it across to one of the dashed boxes that appear when you move the element over the canvas.

You can use the Show catches toggle to show and hide error catches in the palette. A catch is a section of the assembly that is applied when an API call results in the corresponding HTTP status code being returned. Click the catch section to open the property sheet for all your catches.


You can zoom the view of your canvas in and out by clicking the + and - icons. To fit the canvas to the screen size, click the Fit to screen icon .

You can filter the canvas to show only the parts of it that will apply to a specific operation by clicking the Filter by operation icon  and then selecting the operation from the drop-down list. Click the Clear operation filter icon  to remove the filter.

## The property sheet

---

When you select an element that is in the assembly by clicking it, details about the element are displayed in the property sheet. In this pane, you can configure the element's properties. The options available to you in the property sheet are specific to the type of element you are working with. For some elements, you can add and remove properties by clicking Object Properties and selecting the property from the drop-down menu.

You can release the property sheet by clicking the Close icon .

## The behavior of an assembly

---

The assembly runs policies in order and acts on different contexts of the API call. When an API call is made, security and rate limits are enforced before the assembly is executed. During the assembly, the flow can branch or be thrown and caught, according to the policies contained in it. The message context can be thought to flow through the assembly, being used and altered by various policies. In addition to the message, other contexts can be accessed and created.

### Security and rate limiting

Before the assembly is executed, security and then rate limits are enforced.

First, security definitions and CORS access control are used to authenticate an API call. Any API Key security definitions are used to identify applications that have subscriptions to a Product containing the API. If a security definition does not allow access, the API call is rejected.

If an application is identified by its client ID or client secret, a rate limit can be enforced based on the Plan or operation called.

### The assembly execution sequence

The assembly is executed in order from the initial, filled, circle to the final, unfilled, circle. However, there is room for branching, when if and operation-switch logic constructs are used, or for the remaining assembly to be ignored when a throw policy is executed.

The message is the context that is acted upon by any policy that isn't otherwise configured. At the beginning of the API call, the message is empty and, at the end of the API call, the message is used as the response.

The request context contains the information that is sent by the API caller and varies with the type of operation called and the configuration of that operation. For example, a GET operation can never have a populated `request.body`, and you can configure an operation to have `request.parameters` (query parameters). The first policy in an assembly acts on the request and produces the first instance of the message. If there are no policies, the request is returned to the caller.

### Managing contexts

Because the message can be overwritten, it can be useful to create and reference new contexts where possible so that they are saved and reusable during the API call.

- Use the map policy to overwrite the message context when you need to execute a policy that only acts on the message.
- Use the request context when you want to use the original request made to the API.
- Use the map, invoke, and proxy policies to create new contexts when you want to save your message.

Note: When you create a new context, unless you are also mapping to the message, the message is overwritten with an empty object.

For example, an invoke policy is the first policy in the assembly and its response overwrites the request as the message. The message is then acted upon by a validate policy, and a map policy then saves the message as a new context, ready for a second invoke policy to overwrite the message without losing the first invoke policy's output.

You can also access contexts outside of the message or your custom contexts, but these cannot be written to. For a list of contexts, see [API Connect context variables](#).

### Branches and catches

Using logic constructs, such as operation-switch or if, you can execute different sections of the assembly when certain conditions are fulfilled. When the assembly branches, the subsection of the assembly contained by the construct is executed in the same manner as a complete assembly. However, contexts are shared with the complete assembly.

When a catch is triggered, either by an error occurring during the execution of a policy, or because a throw policy is encountered, the rest of the assembly flow is ignored. All contexts are shared by the catch being executed and when the end of the catch is reached, the API call is completed. There is no way to return from a catch to the rest of the assembly.

## Related concepts

---

- [Variable references in API Connect](#)

## Related tasks

---

- [Adding elements to your assembly](#)

## Related reference

---

- [API Connect context variables](#)

---

## Adding elements to your assembly

Create your API assembly by using the Assemble view in the API Designer.



## About this task



---

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can use the assembly tool in the UI to create assemblies that are used to manipulate requests made to or responses made by any of your API's operations.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .


If you want to use the Source view to edit the source of your API definition, see [execute OpenAPI extension](#) for more information about how to configure the OpenAPI YAML.

For more information about the use of the assembly tool, see [The assembly editor](#).

## Procedure

---

To add elements to your assembly using the assembly tool, complete the following steps:

1. In the navigation pane, click  Develop, then select the APIs tab.  
The Develop page opens.
2. Click the title of the API definition that you want to work with.
3. Use the Policies view to add an element.

Note: You must specify a version for the policy. To prevent errors due to incompatible versions of a policy between the API and the gateway, API Connect will only publish an API to the gateway if the policy version in the API exactly matches the version that is present on the gateway.

- a. Select the Gateway tab, then click Policies in the navigation pane.
- b. Find the element that you want to add in the palette.
- c. Drag the element onto the canvas; dashed boxes are displayed. Drop the element in a dashed box to insert it into that position in the assembly.

Note:

- Elements are applied in order from the initial, unfilled, circle to the final, filled, circle.
  - Unless an Operation Switch element is used, the whole assembly applies to every operation in the API.
- d. Add and edit properties of the element by clicking the element and using the property sheet.

For some elements, you can add and remove properties by clicking Object Properties and selecting the property from the drop-down list. For information about the properties of policy elements, see [API policies and logic constructs](#). For information about the properties of logic constructs, see [Logic Constructs](#).

- e. Specify a version for the policy.

To add a version, click the Source icon , and complete the **version** section of the policy YAML. For example:

```
execute:
- invoke:
 version: 2.2.0
 title: invoke
...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions. For more information about how to configure the policy OpenAPI YAML, see [execute OpenAPI extension](#).

4. Optional: Repeat Steps [3.b](#) to [3.e](#) for any additional elements that you want to add.
5. Click Save to save your changes.

## Results

---

You have added one or more elements to your assembly.

## Handling errors in the assembly

---

Use the catch section of the assembly to describe the handling of errors thrown during the assembly execution.

## About this task

---

Note:






- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see [Editing an OpenAPI 2.0 API definition](#).
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see [OpenAPI 3.0 support in IBM® API Connect](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

The catch section of the assembly is used to implement an assembly in the instance that an error is thrown during the assembly execution. For example, the assembly could contain a throw element, the API caller could fail to authenticate, or a policy could fail to execute correctly. Each error can be handled with a different catch and each catch can handle multiple status errors.

## Procedure

To create a catch and include elements in it, complete the following steps:

1. In the navigation pane, click  Develop, then select the APIs tab. The Develop page opens.
2. Click the title of the API definition that you want to apply a catch to.
3. Select the Gateway tab, then click Policies in the navigation pane.
4. Set the Show catches toggle in the menu bar of the canvas to the Show position.
5. Click Catches in the canvas, or a Catch icon  if one is displayed. The property sheet for the API's catches opens.
6. To add a default catch that is executed when an otherwise uncaught error is thrown, click Add default catch. Note: If you have a default catch before another catch in precedence, the default catch will activate even when the other catch's error is thrown.
7. To add a new catch, click Add catch.
8. To specify which errors the catch applies to, type the name of a custom error and press Enter, or use the Search errors field to search for the appropriate error.
9. Optional: To remove an error case from a catch, click the corresponding cross.
10. Optional: To change the precedence of your catches, use the Move up  or Move down  icons. If an error case is handled by multiple catches, the catch at the beginning of the list is applied.
11. To add an element to a catch, drag the element over the dashed, gray box that appears in the flow from the Catch icon  for the catch that you want to apply the element to.
12. Click Save to save your changes.

## What to do next

If you added a catch for `ConnectionError`, `SOAPError`, or `OperationError`, you must add the same error to the Stop on error setting for the Invoke policy in your assembly. Otherwise if the error occurs during the execution of the Invoke policy, it is not caught, the policy execution is allowed to complete, and the assembly flow continues. For details on configuring an Invoke policy, see [Invoke](#).

For details of all the errors that can be returned by the assembly and are available to the catch function, see [Error cases supported by assembly catches](#).

## Validating the OpenAPI YAML source

How to validate the OpenAPI YAML source when you're editing an API.

### About this task

When you're configuring an API in the API editor, you can validate the OpenAPI YAML source by clicking Validate. If any swagger parser validation errors are found, an error icon is displayed that you can then click to see the individual errors, and where they occur in the YAML. You can then fix the errors, save the updates, and continue to validate the OpenAPI YAML throughout the API configuration process.

Note: OpenAPI validation does not show gateway or policy errors that will stop your API from being staged or published. This type of validation occurs only when you try to stage or publish your API. For more information, see [Staging an API](#) or [Publishing an API](#).

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

## Procedure


To validate the OpenAPI YAML source, complete the following steps.

1. Open the required API for editing, as described in [Editing an OpenAPI 2.0 API definition](#) or [Editing an OpenAPI 3.0 API definition](#).
2. Click Validate from the menu bar in the header.



3. If any swagger parser validation errors are found, an error icon is displayed on the header menu bar, with the number of OpenAPI errors found displayed next to the icon.

For example, .

4. You can then click the error icon to see a detailed list of the errors, and their location in the YAML. If you're in Form  view, you can click an individual error to navigate directly to the relevant API section in the form.
5. After you fix the error, click Save to save your updates, and then click Validate again.
6. When all of the errors are fixed, the validation returns a success message.

For example,  API has been validated

## Related concepts

- [Testing an API](#)

## Related tasks

- [Activating an API](#)

- [Staging an API](#)

## LDAP authentication

The Lightweight Directory Access Protocol (LDAP) is an internet protocol for accessing and maintaining distributed directory information services over a network. If you rely on LDAP to authenticate users for web applications, take a minute to review the contents of this topic before beginning.

### Programming guidelines

When authenticating with LDAP, observe the following guidelines:

- Use only a LDAP interface -- Users can only be authenticated via a connection to an LDAP server.
- Active Directory -- Use of an Active Directory interface is prohibited, however, LDAP authentication with Active Directory is supported.
- Administrator properties -- The LDAP administrator must have access to all user's LDAP properties.

LDAP attributes read by IBM® API Connect are as follows:

Attribute	Definition
mail	User's SMTP email address.
cn	Used internally to determine user's common name.
sn	Used internally to determine user's last name or surname.
givenname	Used internally to determine user's first name.
Prefix from Search dn	Used as username.

### Using an LDAP Server for User Authentication

Every entry in an LDAP directory server has a distinguished name (DN). The DN is the name that uniquely identifies an entry in the directory. A DN is made up of **attribute=value** pairs, separated by commas. For example:

```
cn=Ben Gray,ou=editing,o=New York Times,c=US
cn=Lucille White,ou=editing,o=New York Times,c=US
cn=Tom Brown,ou=reporting,o=New York Times,c=US
```

Any of the attributes defined in the directory schema can be used to make up a DN. The order of the component attribute value pairs is important. The DN contains one component for each level of the directory hierarchy from the root down to the level where the entry resides. LDAP DNs begin with the most specific attribute (usually some sort of name) and continue with progressively broader attributes, often ending with a country attribute. The first component of the DN is referred to as the Relative Distinguished Name (RDN). It identifies an entry distinctly from any other entries that have the same parent. In the previous examples, the RDN "cn=Ben Gray" separates the first entry from the second entry, (with RDN "cn=Lucille White"). These two example DNs are otherwise equivalent. The attribute=value pair making up the RDN for an entry must also be present in the entry. (This is not true of the other components of the DN.)

### Examples

In the following examples, a user search is performed from Base DN. The search is done anonymously, or if authenticated bind is used, an authenticated user is used. Search DN appends the prefix, the given user name, and the suffix. If the prefix used is **{uid=}** and the suffix used is **"**", **uid** becomes the user name attribute. The default search filter used is: **(| (cn={filter}\*) (sn={filter}\*) (mail={filter}\*) (givenName={filter}\*))** and attributes used for prefix are also added to the search filter. In this case, where **{uid=}** is the prefix, when searching for users, the search filter becomes:

```
(| (cn={filter}*) (sn={filter}*) (mail={filter}*) (givenName={filter}*) (uid={filter}*))
```

where **{filter}** is replaced by actual text.

The authenticated bind DN is a user on the external LDAP server permitted to get base DNs and search the LDAP directory within the defined search base. It should also be able to read other user properties and be used if anonymous access to LDAP to get base DNs and to search and get access to user attributes is not allowed. When a search is performed for **Steve**, the LDAP query filter shown in the following example is used and search is done from base DN specified in UI. When the user's DN is returned, the DN and password are used to authenticate the user:

```
(| (cn=Steve*) (sn=Steve*) (mail=Steve*) (givenName=Steve*) (uid=Steve*)).
```

For bind during login calls, the search string used is the prefix. For example, if the prefix is **{uid=}**, the search string used to search for a user during log in becomes: **{uid=Steve}**.

Using the mail attribute as the user name

When you want to use an email address as the user name, (for example *steve@company.com*), you typically use the mail attribute as the prefix **{mail=}**. In this case, use the following search string to perform the search internally (assuming **{mail=}** as prefix):

```
(| (cn=steve@company.com*) (sn=steve@company.com*) (mail=steve@company.com*) (givenName=steve@company.com*) (mail=steve@company.com*))
```

In the previous example, if DN is found, the password is used with the DN to perform a bind. The first result is taken, so be sure to use a unique attribute for username. Next, the LDAP properties for the user are read (**cn, sn, mail, givenName**). If LDAP properties cannot be read using the logged-in user, they are read from the user's DN using authenticated bind credentials. If the user name attribute differs, it is also queried for. For example, if the prefix is **{uid=}**, the **uid** attribute is also read from the user's DN object:

```
(| (cn={filter}*) (sn={filter}*) (mail={filter}*) (givenName={filter}*))
```

Note: The prefix and suffix cannot be used to get a user's DN directly. For example, the following attempt to directly get a user's DN fails:

```
Prefix: "uid=_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_con_ldap_requirements_
Suffix: ",ou=users,dc=company,dc=com"
```

## LDAP referrals

---

LDAP referrals allow a directory tree to be partitioned and distributed between multiple LDAP servers. An LDAP referral is a domain controller's way of indicating to a client application that it does not have a copy of a requested object, while giving the client a location that is more likely to hold the object. The client then uses the object as the basis for a DNS search for a domain controller.

API Connect support for LDAP referral includes:

- Searching for users that are part of multiple Active Directory trees and forests.
- Authenticating users in the Cloud Manager, API Manager and the Developer Portal.

Note: API Connect LDAP referral support is dependent on the following conditions:

- A single LDAP host/port is configured with administrator credentials and all users are referred to from the tree/server's base DN.
- As part of the user search following the LDAP referral, the same administrator credentials are used in the downstream trees/forests.
- LDAP API authentication does not support external LDAP referral. Internal LDAP referral is supported.

---

## Authentication URL user registry

You can use an Authentication URL user registry to specify a REST authentication service to manage user authentication, and to optionally provide additional metadata to be embedded in the token.

This support can optionally enable any of the following:

- Providing the authenticated credential to IBM® API Connect. For example, the user logs-in with user name: `spoon`, and password: `fork`. When the user is authenticated, the credential becomes `cn=spoon,o=eatery`. The credential is kept in the OAuth `access_token` to represent the user.
- Providing metadata support. Allow extra metadata to be stored in the `access_token`.
- Overriding the `scope` that the application receives after a successful OAuth protocol processing. By responding with a specific header, the Authentication URL endpoint can replace the `scope` value that the application receives. For example, you can provide a specific resource owner an account number within the `scope` header response for use in future processing steps.

When you call the Authentication URL user registry, the API Connect gateway sends a GET request with HTTP headers and then processes any HTTP response from the URL. For authentication, a REST authentication service is expected at the Authentication URL.

The following response from the REST authentication service indicates that user authentication is successful and that API Connect will use `cn=spoon,o=eatery` as the user identity.

```
HTTP/1.1 200 OK
Server: example.org
X-API-Authenticated-Credential: cn=spoon,o=eatery
```

For information on how to configure a User Security policy in an API assembly for use with an Authentication URL user registry, see [User Security policy](#).

For an example of an OAuth provider configuration that uses an Authentication URL user registry, see [Example - using multiple OAuth policies in an OAuth provider assembly](#).

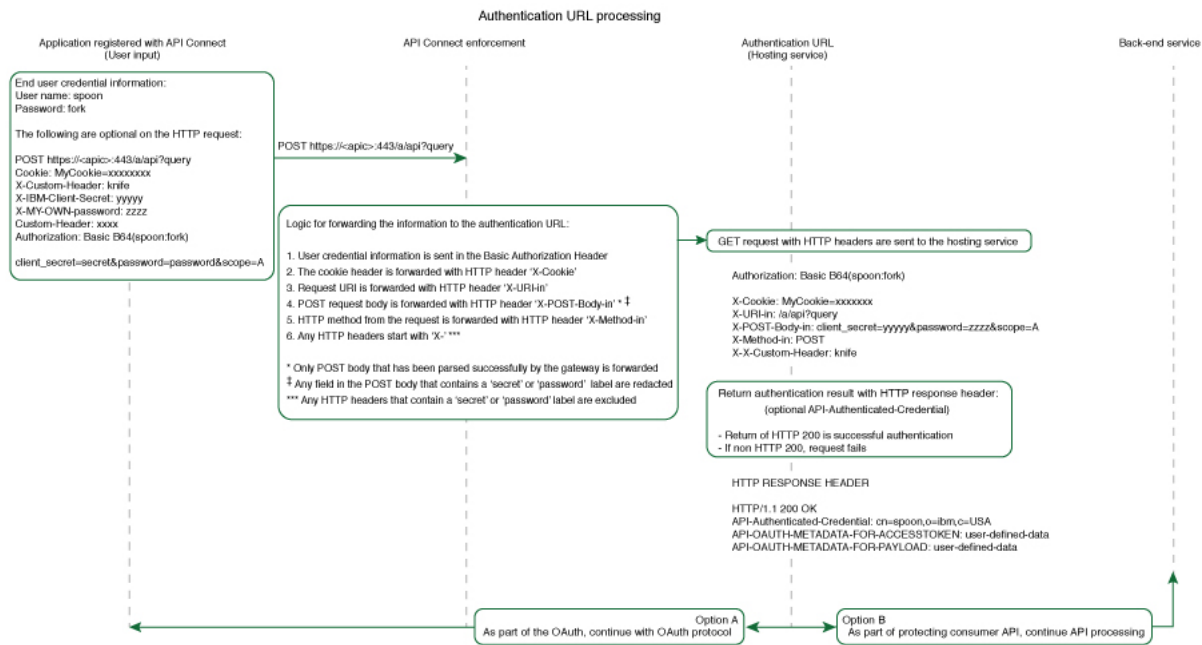
API Connect considers any non-200 HTTP response code a failed user authentication attempt.

When an Authentication URL user registry is invoked, two HTTP response headers are available that include metadata in the access token or the response payload that contains the access token. For more information, see [OAuth external URL and authentication URL](#). The two metadata response headers are:

```
API-OAUTH-METADATA-FOR-ACCESSTOKEN
API-OAUTH-METADATA-FOR-PAYLOAD
```

When an Authentication URL is invoked, an HTTP response header is available to override the requested `scope` from the application. For more information, see [Scope](#). The response header is:

```
x-selected-scope
```



If you are using the DataPower® API Gateway rather than the DataPower Gateway (v5 compatible), this diagram is provided for guidance only and is not fully accurate for this release.

## Activating an API

After you have created an API definition you can activate it to make it available for testing.

### About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

When you activate an API, API Connect automatically completes the following actions:

- Creates a draft Product, adds the API to the Product, and publishes the Product to the Sandbox Catalog so that the API is available to be called. The Product has the title *api\_title* auto product. Note that if you later want to delete the draft Product, you cannot delete it directly; instead, delete the API and the draft Product is deleted together with the API; see [Deleting an API definition](#). If you want to remove the Product from any Catalogs to which it is published, you must do this separately; see [Removing a Product from a Catalog](#)
- Subscribes the Sandbox test application to the Product so that you can immediately test the API in the test environment. For information on testing an API, see [Testing an API](#).

To activate an API, you must be assigned a role that has the **Product:Manage** and **Subscription:Manage** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see [Creating custom roles](#).


Note:

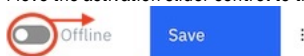
- The API activation will not complete successfully if lifecycle approval is enabled in the Sandbox Catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to activate an API they must be disabled; for information on lifecycle approval settings, see [Creating and configuring Catalogs](#).
- To activate an API from the API Designer user interface, you must be connected to a Management server; API activation is not available with API Designer in offline mode. For more information, see [Logging in from API Designer](#).
- Products that contain an API with a Swagger property using **regex** that include lookahead assertions, such as "(?)" cannot be validated or published. An error message is returned. For example:

```
Product has not been published!
The multipart 'openapi' field contains an OpenAPI definition with validation errors.
definitions.properties.pattern Does not match format 'regex' (context: (root).definitions.properties.pattern, line: 0,
col: 0)
400
```

## Procedure

To activate an API, complete the following steps:

1. In the navigation pane, click  Develop, then select the APIs tab.
2. Click the title of the API that you want to work with.
3. Move the activation slider control to the on position:



## Results

On successful completion, the API is shown as Online:



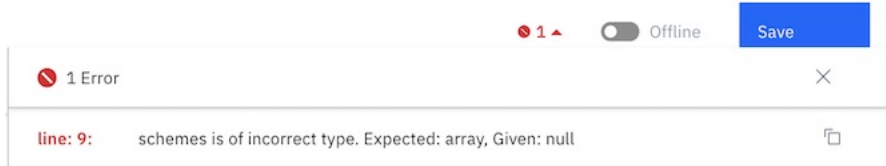
You can stop the API by moving the activation slider control to the off position:



If you stop an API, the application subscription is deleted, and the auto Product is removed from the Sandbox Catalog.

If you make a change to the API, it is republished automatically. You can also republish a running API manually by stopping it and then re-activating it.

The error indicator shows whether there are validation errors in the OpenAPI source for the API definition. If there are errors, click the icon for more details:



You can also activate an API during the creation process, and on the API test page; see [Creating an API definition](#) and [Testing an API](#).

Note: If an OpenAPI 3 API contains a response wildcard (which is not supported), publishing is disabled for that API. You must correct the problem before you can publish the API.

## Testing an API

Select a test tool and verify that your API works as you intended.

Before you publish an API where customers can access it, you need to test it and ensure that it is defined and implemented correctly. API Connect offers tools for running both simple and complex tests, in different environments. Use the following list of options to choose the test tool that best meets your needs:


- Execute and debug the API in the [Test tab](#)  
On the Test tab, you can quickly invoke your API from the API Connect UI with minimal preparation. The Test tab offers full control over the API call through a simple set of fields where you can define your request. The response includes both a visual flow diagram and a code trace that shows how the API's process flow is executed. Note: the Test tab tool is only available to APIs that use the DataPower API Gateway.
- Call the API externally from the Automated API behavior testing application as explained in [Testing an API with Automated API behavior testing](#)  
Automated API behavior testing provides a simple set of fields where you can quickly invoke your API. You can automate tests and define schedules for running them. You can even set up API monitoring and receive alerts on your API health before issues occur.
- Call the API from a [local test environment](#)  
The Local Test Environment is a lightweight API Manager, running on your computer, that allows you to rapidly test APIs without connecting to an API Connect management server.
- Run a simple test in the [Policies editor](#)  
Note: The Test panel is deprecated and might be removed without warning.
- Use the Try it tool in the [Explorer tab](#) to run simple tests  
You can use the Explorer tab in the API editor to see how your APIs look to a consumer in the Developer Portal. You can check the descriptions of the different artifacts, and review any schemas or examples, and you can also use the Try it tool to test the behavior of the API.
- [Locating API information on the Endpoints tab](#)  
The assembly tool's Endpoints tab opens a page that displays information for use when you call APIs externally. This tab displays only when an API's status is Online (the API was activated).
- [Using the Test tab to debug your API](#)  
Use the Test tab to execute calls and trace the API's actions in API Connect.
- [Testing an API with Automated API behavior testing](#)  
Use the IBM Automated API behavior testing application to invoke an API from outside of the API Connect Assembly page so that you can verify that it executes properly.
- [Testing an API with the Policies editor](#)  
IBM API Connect provides a basic test environment in the Policies editor so that you can ensure that your APIs are defined and implemented correctly.
- [Testing an API with the Local Test Environment](#)  
Use the Local Test Environment to test APIs on your local machine, without the need to connect to an API Connect management server. The Local Test Environment is a lightweight API Manager running on your local machine. It allows you to rapidly test APIs locally.
- [Testing an API with the Explorer tab](#)  
Use the Explorer tab in the API editor to see how your APIs look to a consumer in the Developer Portal. You can also use the Try it tool to test the behavior of the API.

## Locating API information on the Endpoints tab

The assembly tool's Endpoints tab opens a page that displays information for use when you call APIs externally. This tab displays only when an API's status is Online (the API was activated).

Use the Endpoints tab to retrieve information that you can use when you call the API from an application.

Note: The fields in the Sandbox Sample Application Credentials section apply to the assembly tool's built-in client application and not to any application that you created. The credentials that display here are the default credentials that display in the assembly tool Test panel's "Identification" section.

To copy information from the Endpoints tab, click  next to a field.

The following values for the API are displayed in the Endpoints tab so that you can copy them and paste them into calls:

**Base API Endpoint**

The URL representing the endpoint of the API that you are editing and testing. To invoke the API, append one of its supported queries (path and parameters) to this URL.

**Client ID and Client Secret**

The values of the built-in test application's client ID and client secret. You can invoke APIs externally by providing the application's credentials in the request header.

**OAuth Token URL**

If you specified an OAuth provider in the API's security definition, then this field displays the URL where you can obtain an access token for calling the API.

**OAuth Auth URL**

If you specified an OAuth provider in the API's security definition, then this field displays the URL where you can submit an access token and receive an authorization token in exchange. You then use the authorization token when calling the API.

**Rate Limit**

If you specified a rate limit on the Security panel when you created the API, the limit is displayed here for reference. If you did not specify a rate limit, the default rate limit displays.

---

## Using the Test tab to debug your API

Use the Test tab to execute calls and trace the API's actions in API Connect.

The Test tab not only lets you quickly invoke an API with any needed headers and parameters, but lets you view the content of the response in both parsed and raw format. Additionally, you can view a trace of the actions execute as the API's process flow operates to complete the call. Viewing the trace can help you debug problems with the API's execution.

Note: The Test tab tool is only available to APIs that use the DataPower API Gateway.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

Complete the following tasks to prepare your API for testing, execute requests, and debug responses.

1. [Preparing an API for debugging with the Test tab](#)

Ensure that your API's definition meets the requirements for debugging it in the API Connect Test tab.

2. [Specifying the testing preferences for an API](#)

By default, when you test an API with the Test tab, a range of test parameters are preconfigured. For example, a default Product is automatically generated, to which the API is added, and that Product is published to the Sandbox Catalog. However, you can configure testing preferences for each of your APIs, including the required Product and target Catalog.

3. [Sending the API request](#)

Fill in fields to set up the request for an API that you want to debug on the Test tab.

4. [Reviewing the API response and trace](#)

Review the response to an API that was invoked in the API Connect Test tab. You can use the included trace information to debug the API's execution.

5. [Testing a GraphQL API with the Test tab](#)

For a GraphQL API definition, the Test tab contains a GraphiQL editor. You can use the GraphiQL editor to help construct a GraphQL query. When you test your GraphQL API, the response is displayed in the GraphiQL editor.

---

## Preparing an API for debugging with the Test tab

Ensure that your API's definition meets the requirements for debugging it in the API Connect Test tab.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To invoke your API in the Test tab, the following requirements must be satisfied:


The DataPower API Gateway is configured to support the API probe

To enable the Test tab's trace feature, the DataPower API Gateway must be configured to support gateway peering and the gateway probe. In a gateway service running in Kubernetes, the API probe is automatically enabled. In a gateway service running external to Kubernetes, the DataPower configuration must be added to the application domain supporting API Connect. Check with your administrator to confirm that gateway peering and the gateway probe are enabled as explained in [Configuring DataPower API Gateway](#).

The API uses the DataPower API Gateway

A gateway exposes APIs to calling applications, and provides processing actions that enable the APIs to integrate with various endpoints. API Connect supports two versions of the DataPower Gateway, but you can only debug APIs that use the DataPower API Gateway. The V5-compatible gateway is not supported. If your API definition has CORS enabled but you cannot see the Test tab, the API is probably configured to use the wrong version of the gateway.


You can determine which gateway your API uses by completing the following steps:

1. Log in to API Manager.
2. In the navigation list, click  Develop, then select the APIs tab.
3. Click the title of the API you want to test.
4. On the API's Design page, locate the Gateway field and note which option is selected.

You can change the gateway selection for an API on the Design page, but you must ensure that the policies used in the API's process flow are compatible with the new selection. For information on changing the gateway for an API, see [Specifying the gateway type for an API definition](#).


CORS is enabled in the API definition

Cross-Origin Resource Sharing support ensures that the API can be accessed from another domain. CORS support must be enabled for your API to be properly invoked and traced. If your API uses the API Gateway but you cannot open the Test tab, it's possible that CORS was not enabled in the API definition. You can enable CORS for you API by completing the following steps:

1. Log in to API Manager.
2. In the navigation list, click  Develop.
3. On the Develop page, click the name of the API you want to test.
4. On the API's Design page, locate the CORS field and select it to enable support.
5. Click Save in the page header.

The API is published in the Sandbox catalog with the default Product/Plan, or a specified catalog and Product/Plan

By default, when you test an API with the Test tab, a default Product/Plan is generated for the API and published to the built-in Sandbox Catalog. However, you can optionally specify a range of API testing preferences including an alternative Product/Plan, and target Catalog; for more information, see [Specifying the testing preferences for an API](#). When you publish (activate) an API, the API is enabled online and becomes available for use. You cannot execute an offline API, even for testing purposes. You can set an API online by completing the following steps:

1. Log in to API Manager.
2. In the navigation list, click  Develop.
3. On the Develop page, click the name of the API you want to test.
4. In the Design page header, click Offline to toggle the API to its Online state.

**Next topic:** [Specifying the testing preferences for an API](#)

---

## Specifying the testing preferences for an API

By default, when you test an API with the Test tab, a range of test parameters are preconfigured. For example, a default Product is automatically generated, to which the API is added, and that Product is published to the Sandbox Catalog. However, you can configure testing preferences for each of your APIs, including the required Product and target Catalog.

---


### About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

Note: This feature is not available if you are using the [Local Test Environment](#).

---

### Procedure

1. Open the required API for editing, in either of the following ways:
  - During the initial creation of an API, the API wizard guides you to enter the minimum configuration settings; on completion of the initial configuration, click Edit API.
  - To open an existing API for editing, complete the following steps:
    - In the navigation pane, click  Develop, then select the APIs tab.
    - Click the title of the API version that you want to work with.



2. Click the settings icon  
The Preferences page opens.
3. Use the Edit links to configure the following testing preferences, as required:

#### Target catalog

The Catalog to which you want to publish the Product that contains the API. The selected Catalog is used as the scope for the other preference settings. You can select a Catalog of your choice, or you can opt to use the Sandbox Catalog by selecting Use the default built-in Sandbox Catalog.

#### Target Gateway service

The gateway service to which you want to publish the Product that contains the API.

#### Target product

The Product that will contain the API; you can select any of the Products from the selected Target catalog, or you can opt to use an automatically generated Product by selecting Generate Product on Save. The auto-generated Product has the title *api\_title* auto product.

#### Target product rate limit

If you opted to use an automatically generated Product, you can configure a rate limit to control calls to the API. If you selected your own Product, the Target product rate limit option is not available; instead, the rate limits that are configured in the selected Product and Plan are applied.

#### Target Plan

The Plan in the selected Product that you want the test application to subscribe to. If you opted to use an automatically generated Product, the Target Plan option is not available; instead, the application is subscribed to the Default Plan in the automatically generated Product.

#### Test application

The application that you want to subscribe to the selected Plan. You can select an application of your choice or, if you opted to use the Sandbox Catalog, you can instead use the pre-supplied test application by selecting Use the built-in Test Application.

#### Test Consumer Org

The consumer organization that you want to subscribe to the selected Plan. If you selected your own application, the consumer organization is preset to that which contains the application, and cannot be changed. If you opted to use the pre-supplied test application you can, if required, select a consumer organization of your choice.

4. Click Save Preferences when done.

---

### Results



The next time you bring the API online, the API is published as specified by the preference settings, ready for testing.

Note: These settings are stored in your browser's local storage and will be used for your API testing in subsequent sessions. The following settings are specific to each individual API:

- Target product
- Target product rate limit
- Target Plan

The remaining settings will be applied to all your APIs.

**Previous topic:** [Preparing an API for debugging with the Test tab](#)

**Next topic:** [Sending the API request](#)

---


## Sending the API request

Fill in fields to set up the request for an API that you want to debug on the Test tab.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

---

## Open the Test tab

1. Log in to API Manager or API Designer.
2. In the navigation list, click  Develop, then select the APIs tab.
3. Click the title of the API you want to test.
4. On the API Design page, click the Test tab in the page header.

---

## Fill in the "Request" section

Use the Request section of the page to set up the request URL, the authentication mechanism, and the request parameters. Complete the following steps to fill in the fields needed for configuring the request.

1. Select an operation and request URL from the list provided.
2. On the Parameters tab, define header, query and path parameters.  
An empty row is provided for you to define parameters; enter the parameter name in the Key field, select query, header, or path in the Located in field as appropriate, and supply a string value in the Value field. As you begin to define a new parameter, a further empty row is added automatically.

Default header parameters and values are provided. For example, if the API has client ID or client secret definitions applied, the corresponding keys are added as header parameters, with the values preset; for details on configuring client ID and client secret definitions, see [Creating an API key security definition](#).

The following table lists some common header definitions that are used in a request:

Table 1. Common request headers

Key	Value	Comments
Accepts	<i>media_type/sub type</i>	Specify the type of content that the response headers should use. The default is <b>application/json</b> .
X-IBM-Client-Id	<i>client_ID_val</i>	If you are using the built-in test application in the Sandbox catalog, the value is pre-populated automatically. If you are using your own client application, replace the value with the client ID of your application.
X-IBM-Client-Secret	<i>client_secret_val ue</i>	If you are using the built-in test application in the Sandbox catalog, the value is pre-populated automatically. If you are using your own client application, replace the value with the client secret of your application..
Authorization	Bearer <i>access_token</i>	Encode the token in base64. API Connect cannot encode the token for you.
Content-Type	<i>media_type/sub type</i>	Specify the type of content that the response body should use; for example, <b>application/json</b> or <b>image/png</b> .

If the API has path or query parameters defined, these are added to the parameters list, ready for you to supply values; for details on configuring path and query parameters, see [Defining Paths for a REST API](#) and [Configuring an operation](#).

3. If your API definition uses Basic authorization for the security setting, click the Authentication tab and provide the User name and Password needed for authenticating with the user registry.  
When you invoke the API, API Connect populates a header using the provided information.  
If your API definition does not use Basic authorization, the Authentication tab is not available.
4. If the operation is POST or PUT, set up the request body.
  - a. Click the Body tab.
  - b. Type the information for the body of the request.

---

## Send the request

When your API request is configured, click Send to execute the call.

If a message displays indicating "No Response", the API call cannot be completed. Possible causes of this problem are as follows::

- CORS is not enabled in the API's definition  
Edit the API's definition and enable CORS. Then, save the change and publish the API. If you are using your own client app, remember to subscribe it to the API again after publishing.
- The gateway service URL has an invalid certificate

Follow the instructions to accept the certificate and continue testing.

- The browser cannot connect to the gateway service

**Previous topic:** [Specifying the testing preferences for an API](#)

**Next topic:** [Reviewing the API response and trace](#)

---

## Reviewing the API response and trace

Review the response to an API that was invoked in the API Connect Test tab. You can use the included trace information to debug the API's execution.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

When the API request is invoked, the response displays in the "Response" section of the Test tab. The response always includes the HTTP status code the amount of time it took to receive a response. If the call completes (even if it returns an error), the response also includes headers and a body.

Click the Body tab to see the body of the response. When you review the body of the response, click the Parsed tab to view the response using the format that you specified in the "Accepts" request header. Click the Raw tab to view the unformatted response body.

Click the Headers tab to see the content of request headers and response headers.

Attention: If the assembly or back-end API returns a Cache-Control header, add the **Cache-Control: no-store** request header to prevent the browser from caching the request. Depending on your server, you might want to use the **Cache-Control: no-cache** header instead.

Click the Trace tab to display a record of the API execution so you can see what actions were triggered and the code that we executed for each action.

---

## Examine the trace

Use the Trace tab to see exactly how the API call was executed. This is helpful when the call returned an error and you don't know why. You can debug the API by reviewing the trace to see each step of the API's execution.

Note: The Trace works only with the default Gateway's URL. You cannot select a different Gateway for the trace.

The Trace tab contains the following components to help you debug the API:

Process flow diagram

In the diagram, the policies (actions) that were executed during the call are highlighted while the rest of the process flow is dimmed. The highlights make it easy to see where the executed actions occurred in the overall flow while clarifying the actual flow of execution for the call. For example, if the process flow includes a switch with three options, only the option that was selected during the call is highlighted.

Policies list

A list of the highlighted policies lets you select a policy to examine its trace.

Advanced toggle

The Advanced toggle that lets you control the level of detail in the trace. A basic trace shows the code for the input and output of the selected policy. An advanced trace shows the full code for the policy's execution.

Code box

The selected level of code for the current policy displays in the code box, where you can review it in detail to see exactly how that policy was executed. By default, a minimal version of the response displays (the endpoint, the request, and the response, and the status message), with sections collapsed. You can expand individual sections by clicking the expand icon next to each. To see the complete response, click the Advanced toggle.

**Previous topic:** [Sending the API request](#)

**Next topic:** [Testing a GraphQL API with the Test tab](#)

---

## Testing a GraphQL API with the Test tab

For a GraphQL API definition, the Test tab contains a GraphQL editor. You can use the GraphQL editor to help construct a GraphQL query. When you test your GraphQL API, the response is displayed in the GraphQL editor.

You can test a GraphQL query in either of the following ways:

- Use a **POST** operation, in which case the query is sent in the body of the operation request.
- Use a **GET** operation, in which case the query is sent as a query parameter.

For each operation type, you can test the query against either the GraphQL API endpoint itself, or against the cost endpoint that returns details of the cost of a request to the GraphQL API endpoint.

Whichever method and endpoint you use, you can either supply the query directly, or you can use the GraphQL editor to help construct the query; the editor provides type ahead assistance based on the schema that is defined for the API, and highlights errors along with correction suggestions. The schema was initially uploaded when the GraphQL API definition was created, but might have been modified subsequently. For details on creating a GraphQL API definition, see [Creating a GraphQL proxy API](#). For details on modifying the schema, see [Using the GraphQL schema editor](#).

---

## Before you begin

See [Preparing an API for debugging with the Test tab](#) for the requirements that your API definition must meet for you to test it with the Test tab.

---

## Procedure

1. In the Request section, select the required operation type and endpoint; for example:
  - **POST** `https://myserver.com/myorg/sandbox/mybasepath/graphql`

- GET `https://myserver.com/myorg/sandbox/mybasepath/graphql/cost`
2. On the GraphQL tab, in the left hand pane of the editor, supply the query; for example:

```
{
 accounts(limit: 100) {
 name {
 first
 last
 }
 }
}
```


3. If required, you can supply additional request headers on the Parameters tab.
4. If your query includes variables, you can supply values for the variables in the Query Variables pane. For example, if the query is as follows:

```
query MyQuery($Query__accounts__limit: Int)
{
 accounts(limit: $Query__accounts__limit) {
 name {
 first
 last
 }
 }
}
```

you can set a value for the `Query__accounts__limit` by entering the following in the Query Variables pane:

```
{
 "Query__accounts__limit": variable_value
}
```

where `variable_value` is the required value.

5. Click either Send, or the Execute Query icon  in the editor. The response is displayed in the GraphQL editor, and the Trace section shows you how the API call was executed. For details on the trace information, see [Examine the trace](#).

Previous topic: [Reviewing the API response and trace](#)

---

## Testing an API with Automated API behavior testing

Use the IBM® Automated API behavior testing application to invoke an API from outside of the API Connect Assembly page so that you can verify that it executes properly.

Automated API behavior testing provides a simple set of fields where you can quickly invoke your API. You can automate tests and define schedules for running them.

Automated API behavior testing is an optional add-on to API Connect, and must be deployed by your administrator. If Automated API behavior testing is available in your deployment, a Test API displays on the Home page in API Manager.

Known limitations:

- AutoTest Assist is only available in Cloud Pak for Integration 2022.2.1 and later.
- Insights-Driven Test Generation is only available in Cloud Pak for Integration 2021.2.1 and later.

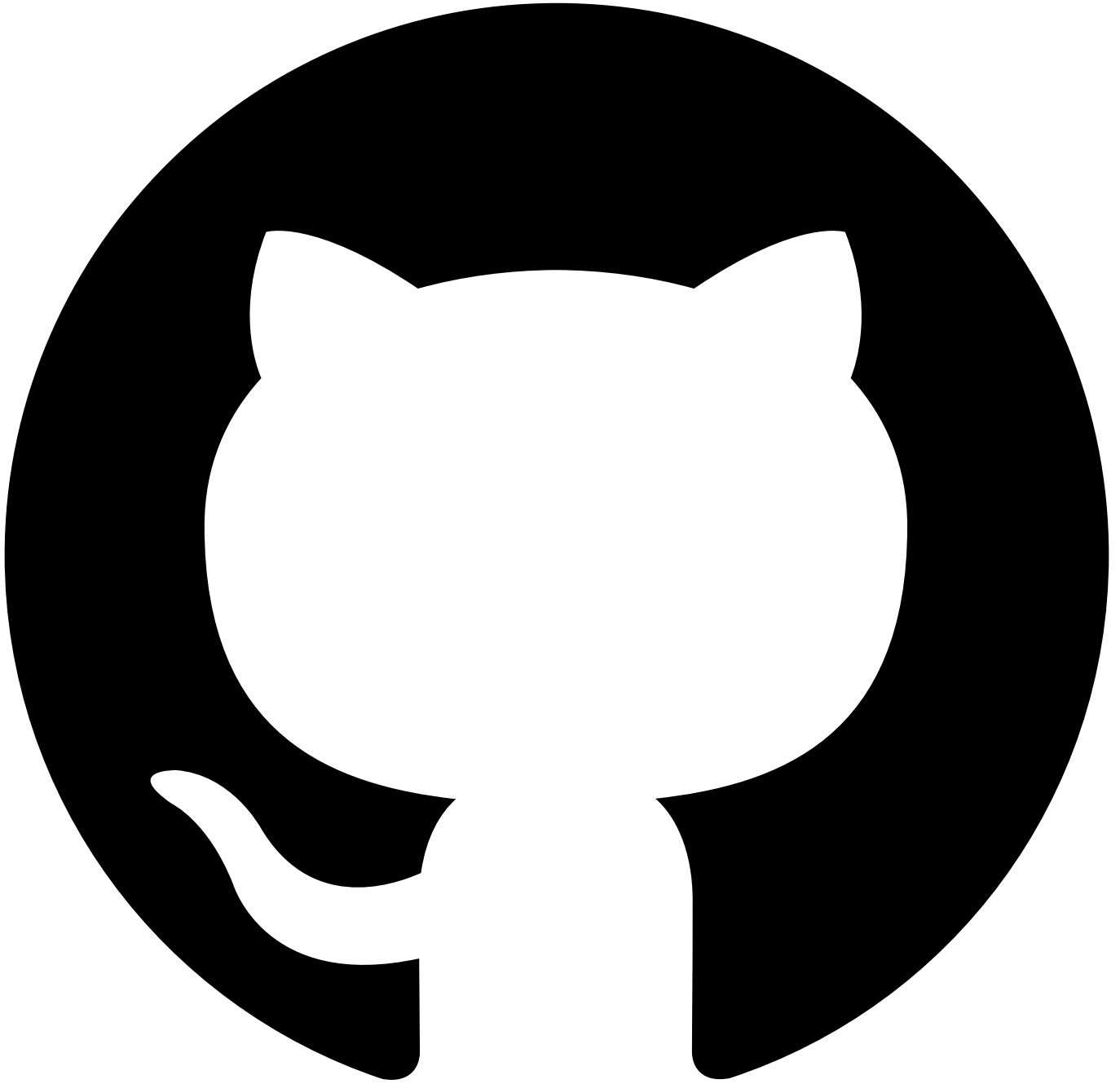
The following OpenAPI 3.0 features are not currently supported in Automated API behavior testing, but their usage doesn't impact on any other supported features:

- The `links` object; allows developers to describe how various values returned by one operation can be used as input for other operations.
- The `callbacks` object; enables asynchronous, out-of-band requests that your service sends to some other service in response to certain events.
- The `oneOf` keyword for defining complex schemas; allows developers to define more sophisticated validation rules. Automated API behavior testing doesn't generate a valid test for this keyword.

Note that you can add checks for missing assertions manually by using the **Add Request/Assertions** option.

If you want to use Automated API behavior testing for validating an API, complete the following steps to get started.

1. Log in to API Manager.
2. Publish your API to make sure that it's available online.
3. Create a client application and copy its client ID and client secret.
4. Subscribe the client app to the API.
5. Open the Test application.
  - Cloud Pak for Integration only: Click Design in the page header, and then click Test APIs; or
  - All users: Click API Manager in the banner to return to the Home page, and then click the Test API tile.



Contribute in GitHub: [Edit online](#)

---

## Getting started

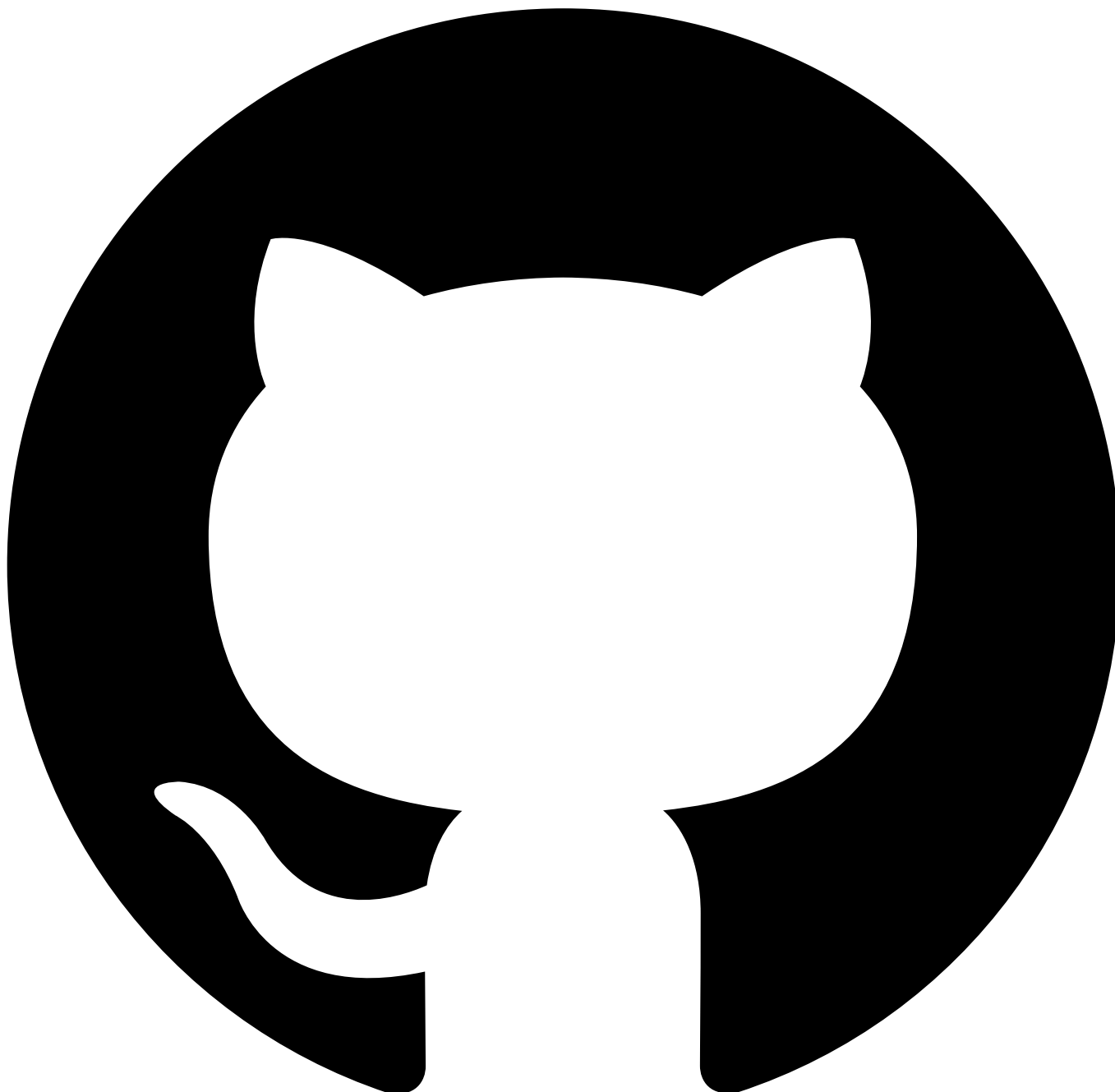
The Automated API behavior testing interface provides collaborative and powerful API testing capabilities.

- **Create automated and simple to use API tests quickly and easily.**
  - Produce high quality APIs consistently, with automated and simple to use API testing that you can rely on.
- **Intelligent validation of API payload accuracy.**
  - Add more intelligence to your tests to verify the accuracy of the payload. You can easily make logical assertions to create flexible and powerful conditions for validation, with no code necessary.
- **Collaborative testing.**
  - Accelerate your API development and testing, by sharing your tests and results through the intuitive visual composer.
- **Easily integrate into your CI/CD pipeline to execute tests as part of your deployment.**
  - Create [API Hooks](#) to allow your tests to be run via the command line and become part of your automated build processes.
- **No code necessary.**

---

## How to get started

- [Creating a test](#)
- [Modifying a test](#)
- [Running a test](#)
- [Publishing a test](#)
- [Variables](#)



Contribute in GitHub: [Edit online](#)

---

## Creating a test

Creating an API test involves the following three steps:

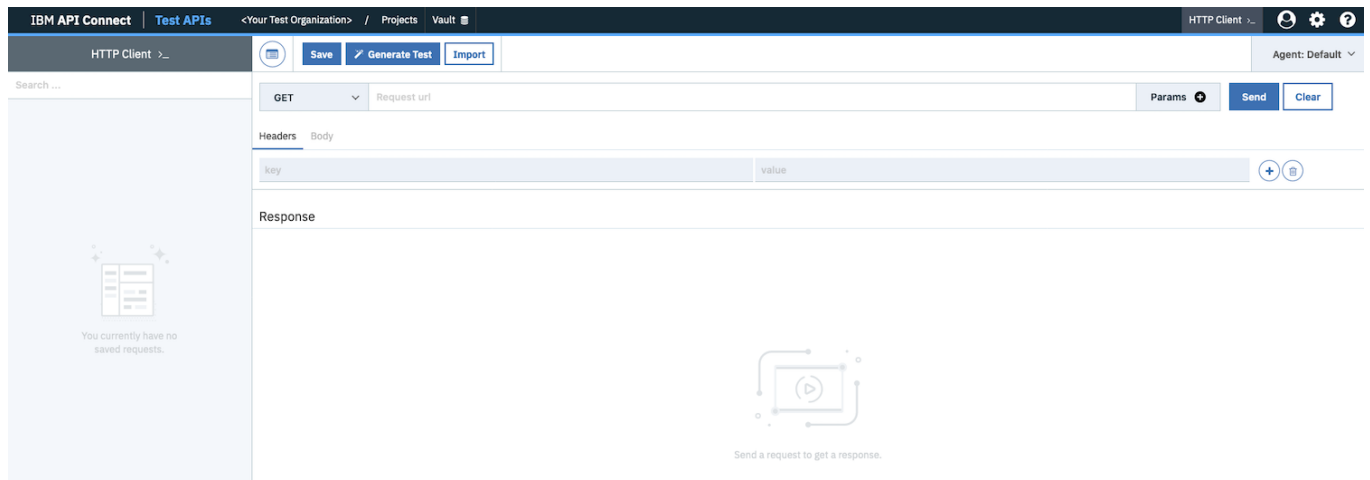
1. Creating a request.
2. Sending the request to an API endpoint.
3. Generating a test from the response that's returned from the endpoint.

The following instructions show you how to create a test by using the **HTTP Client**.

---

## What is the HTTP client?

After you log in to IBM API Connect Test APIs, you land on the HTTP Client.



To reach the HTTP Client from any other page, click **HTTP Client** in the menu bar.

## Step 1: Creating a request

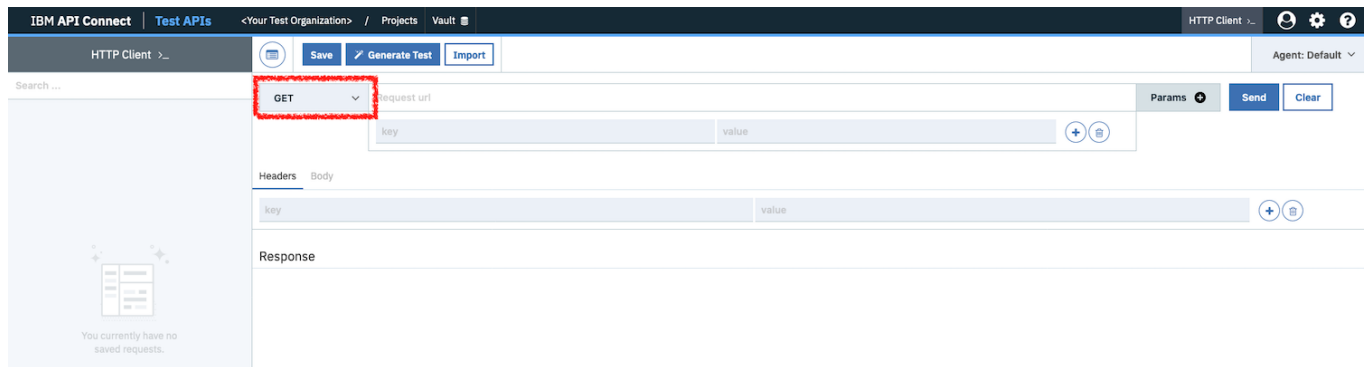
The HTTP Client page contains the following three sections:

1. The initial section is where you specify the request type, the URL, and any other parameters that are needed. It's also where the **Send** and **Clear** buttons are located.
2. The central Headers and Body section is where you define the request headers and body.
3. The Response section is where the headers and body of the response appear after you've sent the request. You don't need to do anything in this section.

The following instructions use **GET https://us-east.apitest.apiconnect.ibmcloud.com/app/api/examples/retail/products** for illustration purposes, which is a built-in API endpoint example. This example requires no parameters to be added to the URL, and requires no request headers or body to be supplied.

## Set the request type

In the initial section of the HTTP Client, select the type of request that you want from the drop-down menu alongside the **Request url** field.



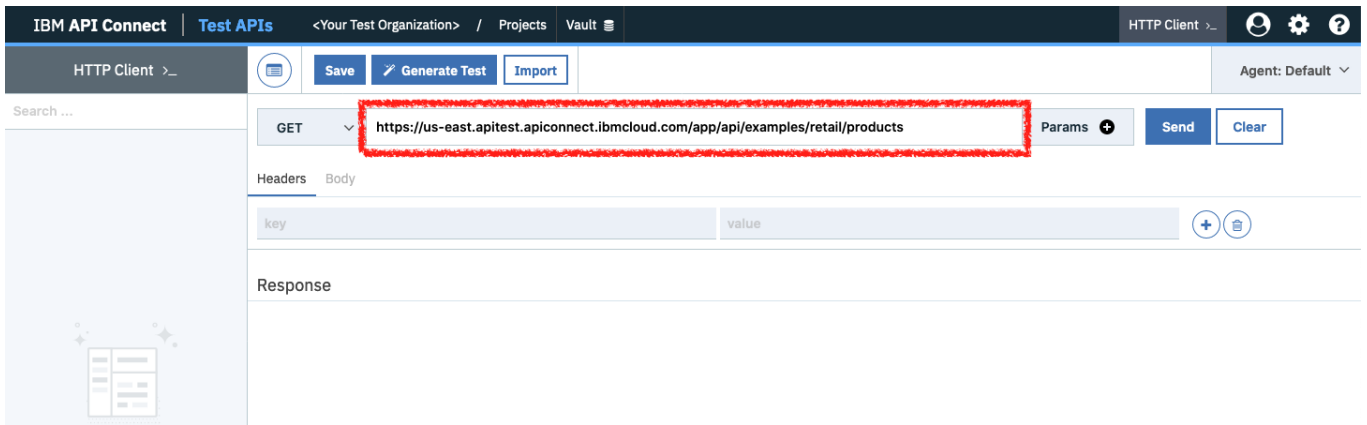
The following options are available:

- **GET**
- **POST**
- **PUT**
- **PATCH**
- **DELETE**

For our example, select **GET**.

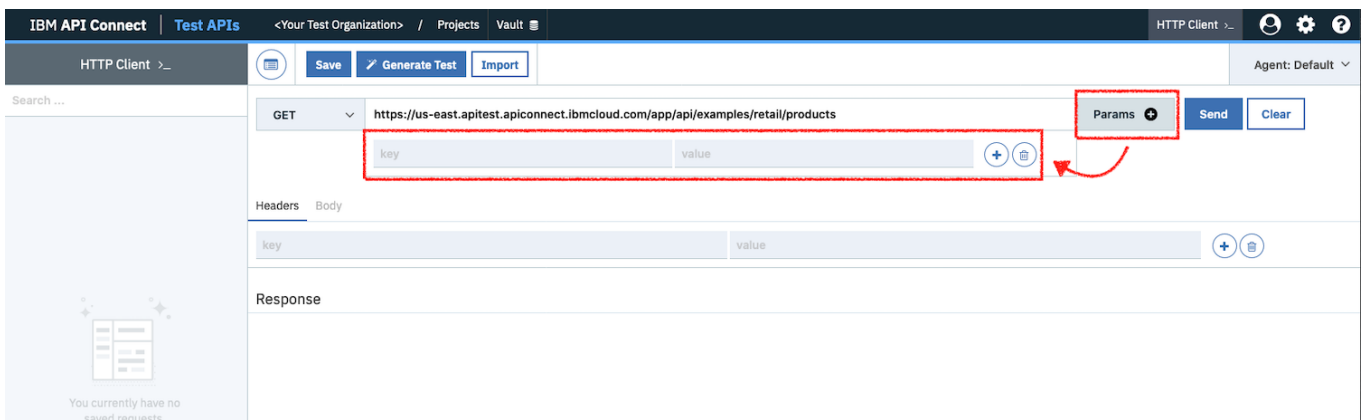
## Set the API endpoint URL

In the initial section, complete the **Request url** field with the API endpoint URL. For our example, enter: **https://us-east.apitest.apiconnect.ibmcloud.com/app/api/examples/retail/products**.



## Add parameters to the URL

If you are using a different API endpoint from our example, and it requires one or more parameters, click **Params +** next to the **Request url** field. A parameter line is added after the URL, which you can edit.



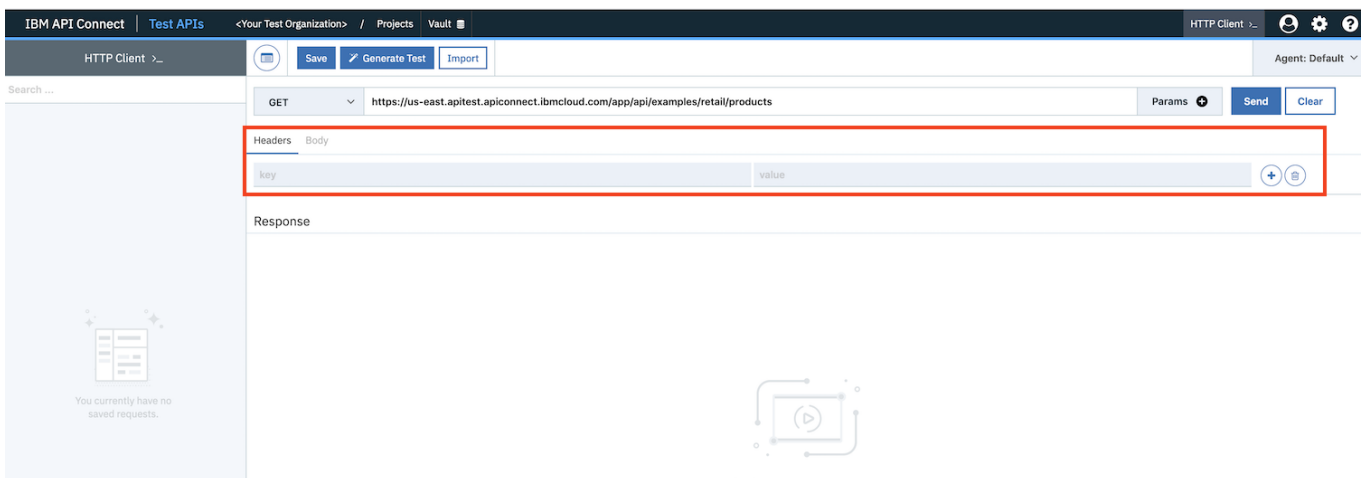
Enter a parameter as a key-value pair.

To add more parameters, click the **add** icon  next to any defined parameter.

To remove a parameter, click the **delete** icon  next to the parameter.

## Configure the request headers

In the central Headers and Body section of the HTTP Client, select the **Headers** tab. If you're using our example, you don't need to supply any headers.



Define a header as key-value pair.

To add more headers, click the **add** icon  next to any defined header.



To remove a header, click the **delete** icon  next to the header.

## Configure the request body

In the central nHeaders and Body section of the HTTP Client, select the **Body** tab. This tab is not available if the request type is **GET**, so you don't need to complete this section if you're following our example.

Set the body type to either **URL Encoded** or **Raw**.

If you set the body type to **URL Encoded**, you must provide one or more key-value pairs.

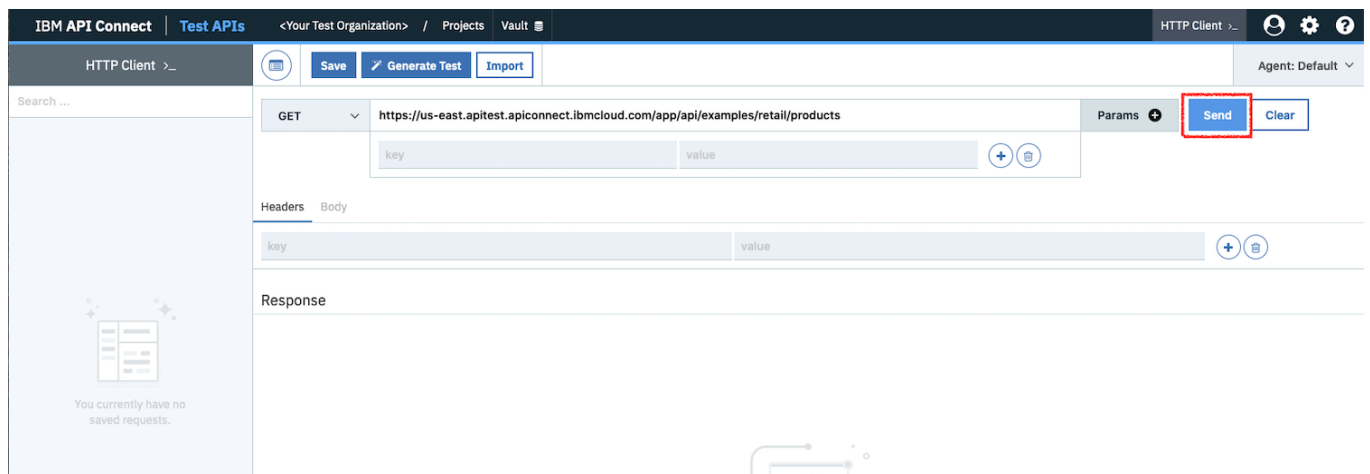
- To add more key-value pairs to the body, click the **add** icon  next to any defined key-value pair.
- To remove a key-value pair, click the **delete** icon  next to the key-value pair.

If you set the body type to **Raw**, you must complete the following actions:

- Select the content type from the content type drop-down menu. The following options are available:
  - `application/x-www-form-urlencoded`
  - `application/json`
  - `text/xml`
  - `text/plain`
- Complete the request body in the text box under the body type selection.

## Step 2: Sending the request

Click **Send** in the initial section of the HTTP Client.



The response from the endpoint is displayed in the Response section of the page. You can examine the returned headers and body by clicking the **Headers** and **Body** tabs in the response section.



IBM API Connect | Test APIs | <Your Test Organization> / Projects | Vault | HTTP Client >...

HTTP Client >... | Save | Generate Test | Import | Agent: Default >

Search ...

GET | https://us-east.apitest.apiconnect.ibmcloud.com/app/api/examples/retail/products | Params + | Send | Clear

Headers | Body

key | value | + | -

Response

Headers | Body | Raw | **Parsed**

● HTTP Code: 200 | Latency: 314 | Fetch: 106 | Total Time: 420

```

Date → Wed, 03 Jun 2020 15:27:44 GMT
Content-Type → application/json;charset=UTF-8
Transfer-Encoding → chunked
Connection → keep-alive
Set-Cookie → JSESSIONID=8AC4BEA4AD09DE362E8B0356D30E6D6A; Path=/app; Secure; HttpOnly
Cache-Control → no-store
Pragma → no-cache
X-Frame-Options → SAMEORIGIN
X-Application-Context → application:docker:8443
Strict-Transport-Security → max-age=31536000; includeSubDomains
Content-Security-Policy → script-src https: 'unsafe-eval' 'unsafe-inline'; object-src 'none'
X-Content-Type-Options → nosniff
X-XSS-Protection → 1
Strict-Transport-Security → max-age=31536000

```

You currently have no saved requests.

## Step 3: Generating a test

Now that you've sent a request and received the response, you can generate the test.

Click **Generate Test** in the initial section of the HTTP Client.

IBM API Connect | Test APIs | <Your Test Organization> / Projects | Vault | HTTP Client >...

HTTP Client >... | Save | **Generate Test** | Import | Agent: Default >

Search ...

GET | https://jservice.io/api/random | Params + | Send | Clear

Headers | Body

key | value | + | -

Response

Headers | Body | Raw | **Parsed**

● HTTP Code: 200 | Latency: 338 | Fetch: 138 | Total Time: 476

```

Connection → keep-alive
X-Frame-Options → SAMEORIGIN
X-Xss-Protection → 1; mode=block
X-Content-Type-Options → nosniff
X-UA-Compatible → chrome=1
Access-Control-Allow-Origin → *
Access-Control-Request-Method → GET,POST,OPTIONS
Content-Type → application/json; charset=utf-8
Etag → "ce54d95f738d2604248212e79a98c816"
Cache-Control → max-age=0, private, must-revalidate
X-Request-Id → 18b451cb-4585-40af-92c0-97c470652617
X-Runtime → 0.024626
Server → WEBrick/1.3.1 (Ruby/2.0.0/2015-04-13)
Date → Wed, 03 Jun 2020 12:53:43 GMT
Content-Length → 490
Via → 1.1 vegur

```

You currently have no saved requests.

Enter a test name.

**Generate Test** ✖ ✓

---

Name

Save to Project

From the **Save to Test Suite** drop-down menu, select a test suite name, or select **Create new test suite** to create your own test suite.

Generate Test ✕ ✓

---

Name

Save to Project Select project ▼

Create new project  
project-1

If you selected **Create new test suite**, enter a name for the test suite.

Generate Test ✕ ✓

---

Name

Save to Project Create new project ▼

my-project

Click the **confirm** icon ✓ to save the test and start the test generation.

Generate Test ✕ ✓

---

Name

Save to Project Create new project ▼

my-project

After generation is complete, the **All set!** page is displayed. This page confirms the creation of your generated test.

Click **Close** to continue.

All set!

---

**Input/Globals**

- ✓ Global params added successfully.
- ✓ Input-set added successfully.

**Assertions**

- ✓ Request added successfully.
- ✓ Assertions added successfully.

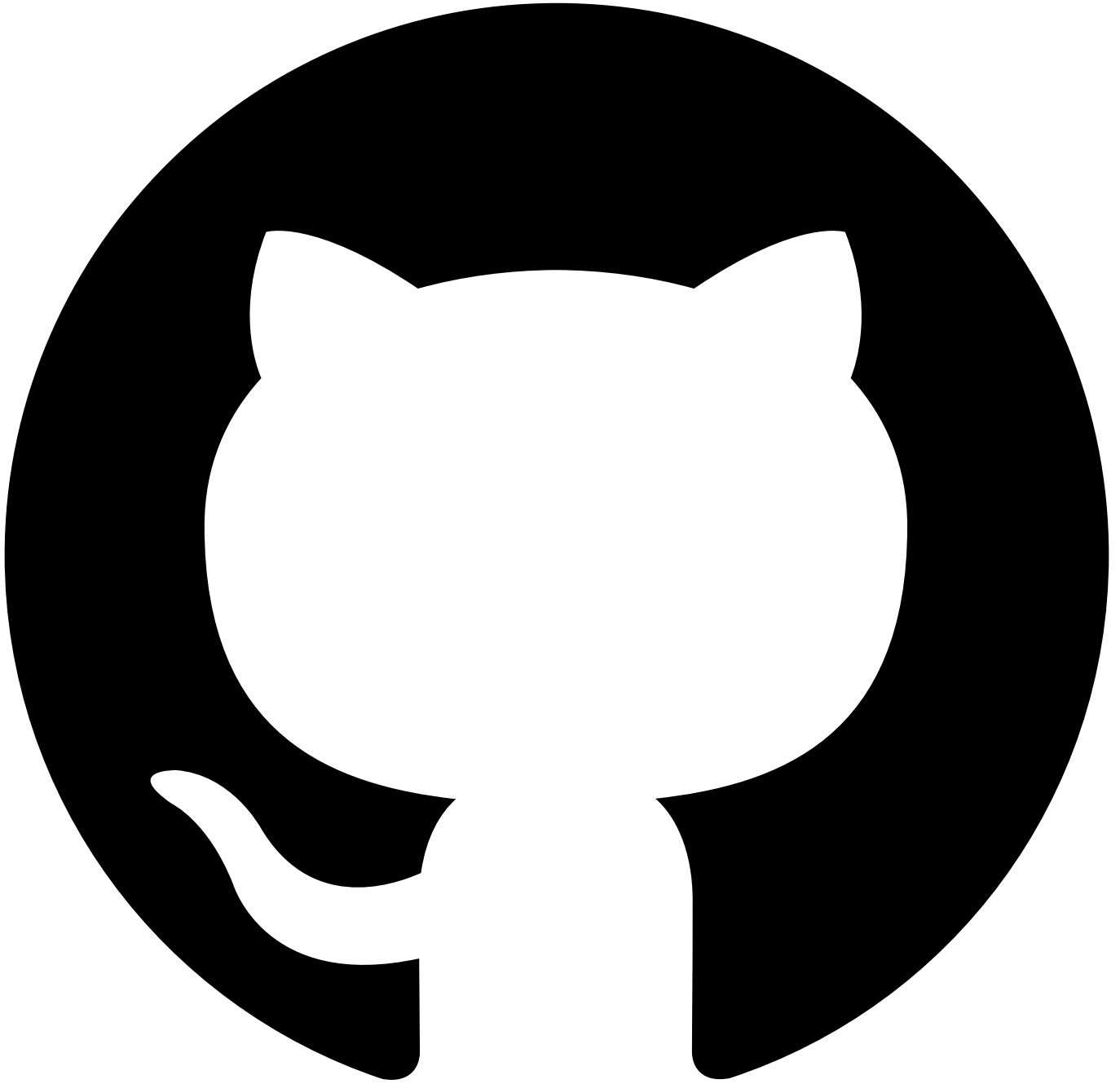
**Close**

You are now in the test editor, called the Composer. In the Composer you can edit and run the test.

The screenshot displays the IBM API Connect interface for configuring a test. The top navigation bar includes 'IBM API Connect', 'Test APIs', and the current project path '<Your Test Organization> / my-project'. The main workspace is divided into a left sidebar with 'Data Sets', 'Vault Beta', and 'HTTP Client' sections, and a main area for the test configuration. The test configuration includes a GET request to 'https://service.io/api/' and five assertions: 'payload\_response.statusCode must be equal to 200', 'payload must match us\_zipcodes', 'payload must be boolean', 'payload must exist', and 'payload must contain id'. Each assertion is color-coded (Aeq, Am, Ai, Ae, Ac) and includes details like mode, type, level, and execution options. A '+ Add Request / Assertions' button is at the bottom.

## What to do next

- Next topic: [Modifying a test](#)



Contribute in GitHub: [Edit online](#)

---

## Modifying a test

Use the Composer to update and modify your test. The following sections provide information about the Composer, and how to use it.

[About the Composer](#)

[Using the editor](#)

[Adding a component](#)

[Editing a component](#)

[Moving components](#)

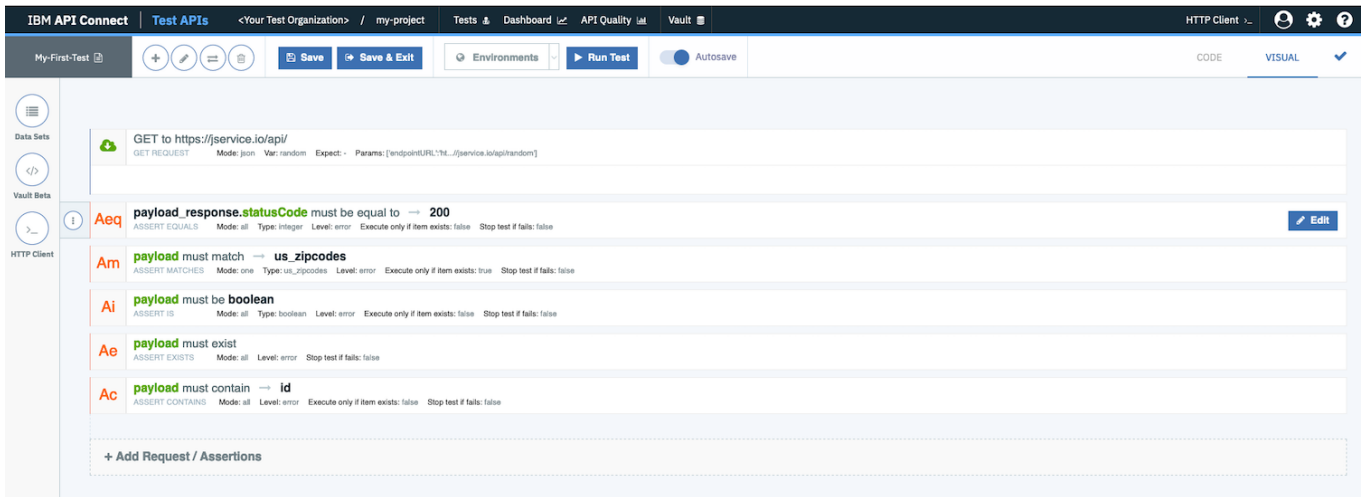
[Transforming a component](#)

[Deleting a component](#)

---

## About the Composer

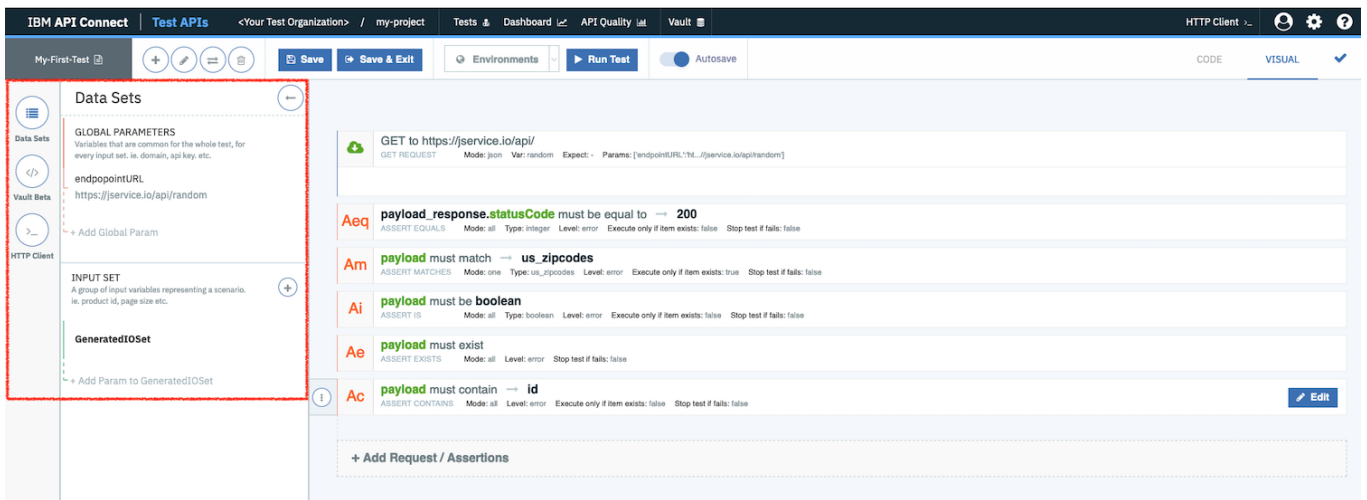
If you are not on the Composer page, or you want to modify a different test, follow the instructions in [Opening a test in the Editor](#) to find the test that you want to modify, and open it in the Composer.



The Composer shows all of the components that make up the test. In our getting started example, the first component is the request to the API endpoint - which in this case is a `GET` to `$(endpointUrl)`. When the test was generated, it extracted the URL that you provided into a variable. To see any variables that were generated, click



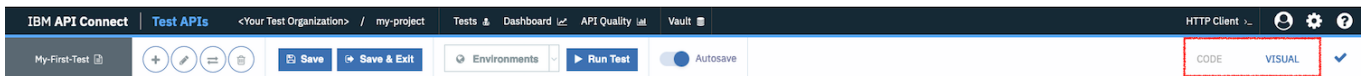
the **Data Sets** icon in the navigation pane.



After the request line, you will see the assertions and other components, for example `Aeq`, `Ai`, `Ae`, `for each in` and so on. For more information about assertion components, see [Assertion components](#).

## Using the editor



The test can be edited by using either the code (text) editor, or the visual (graphical) editor. Toggle between these two editors by clicking the **CODE** and **VISUAL** tabs. The visual editor is the default editor.



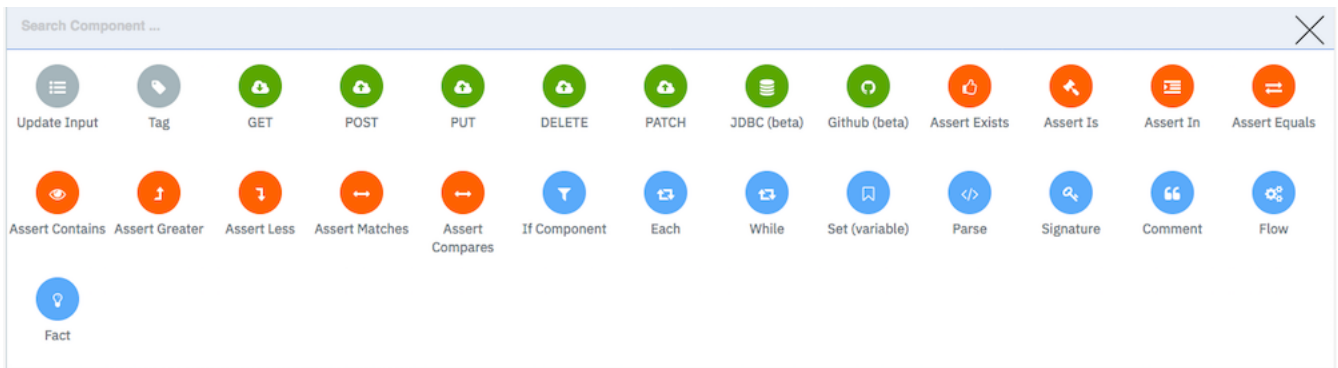
The following instructions describe how to make changes by using the visual editor.

## Adding a component


You can add a component by using one of the following options:

- By clicking the **add component** icon .
- By clicking a component at the location where you want to add a new component, then clicking the **menu** icon , and selecting **Add component before** or **Add component after**, depending on where you want the new component to be positioned.

A list of component types is displayed.





After you've chosen your component type, carry out the following steps to add the component to your test. Note that you can hover over each component to see a description of what that component does.

1. Click the component type that you want to add.
2. Configure the new component's parameters in the component dialogue box.
3. Click the **Confirm changes** icon  to apply the changes.
4. If your new component is in the wrong location, you can drag it to the correct location. Note that you can drag any component in your test.
5. Click **Save** to save your changes.





## Editing a component

You can edit a component in any one of the following ways:

1. Double-click the component to be edited.
2. Click the component, and then click the **Edit component** icon .
3. Click the component, then click the **menu** icon  and select **Edit component**.
4. Click the component, then click **Edit**.

The edit dialogue box is displayed. Each component dialogue box will look different, as they all have different sets of parameters.

In the edit dialogue box, you can make the following changes:

- Modify any of the component's parameters.
- Clone the component by clicking the **Clone component** icon .
- Delete the component by clicking the **Remove component** icon .
- Cancel changes by clicking the **Cancel changes** icon .
- Apply changes by clicking the **Confirm changes** icon .

Click **Save** to save your changes.



## Moving components

Simply drag-and-drop any component from one position within the test to another position in the visual editor.

Remember to click **Save** to save your changes.

## Transforming a component





You can transform a request component to be a different type of request, and an assertion component to be a different type of assertion.

1. Click the request or assertion component to be transformed, then click the **Transform component** icon  .
2. Select the type of request or assertion that you require.
3. Complete the edit dialogue box for the new type of component.
4. Apply the changes by clicking the **Confirm changes** icon  .
5. Click **Save** to save your changes.

---

## Deleting a component

You can delete a component in any one of the following ways:

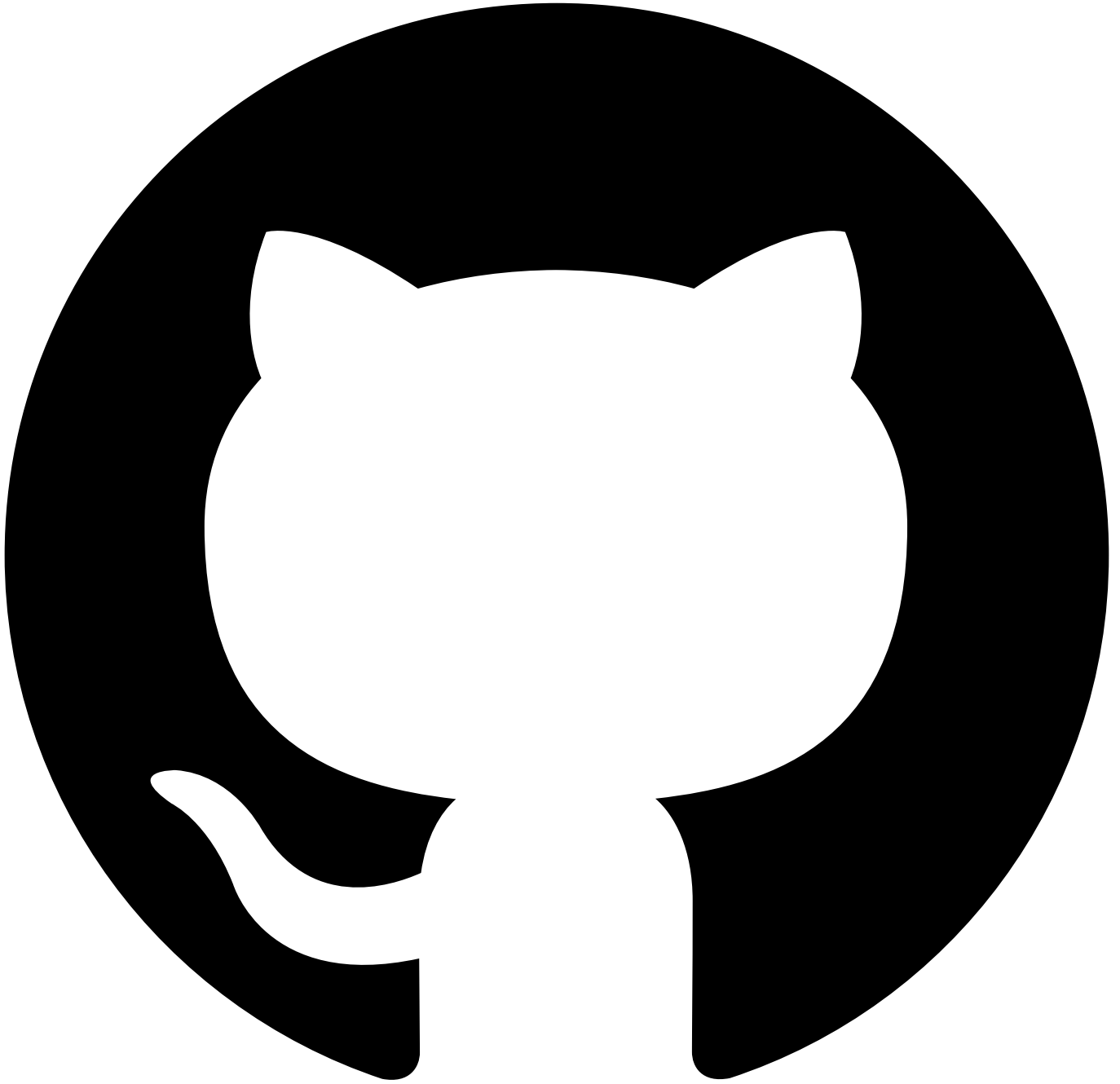
- Click the component to be deleted, and click the **Remove selected component** icon  .
  - This technique can also be used to delete a block of components. Click the first component in the block, and press the shift key while clicking the last component in the block. Then click the **Remove selected component** icon  .
- Click the component to be deleted, then click the **menu** icon  , and select **Remove component**.
- Click the component to be deleted, then click **Edit** alongside the component pane, and then click the **Remove component** icon  .

Remember to click **Save** to save your changes.

---

## What to do next

- Next topic: [Running a test](#)
- Previous topic: [Creating a test](#)



Contribute in GitHub: [Edit online](#)

---

## Running a test

Tests are run from the Composer page of IBM API Connect Test APIs.

If you are not on the Composer page, or you want to run a different test, see [Opening a test in the Editor](#) for instructions.

---

## Before you begin

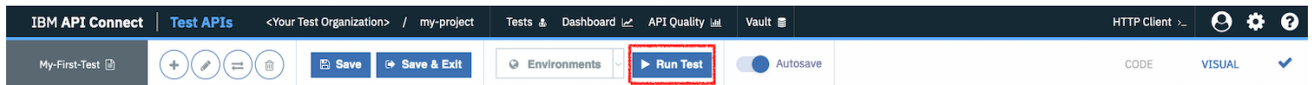
Ensure that your browser is set to allow pop-ups for this application.

---

## Running the test

1. From the Composer page for the test that you want to run, click **Run Test**.





2. Select a downloader from the drop-down list in the "Run Test" dialog box.
3. Click **Run Test**.
4. Your test results are displayed in a Test Report on a separate tab in your browser.

## <Test Organization Name>

### Test Report

<b>Project:</b>	my-project
<b>Event Date:</b>	Mon 11 Feb 2019 04:40 PM +00:00
<b>Test:</b>	
<b>Location:</b>	us-east
<b>Mode:</b>	On request

#### Summary

Response :	<b>Test passed</b>
Failures :	<b>0</b>

#### Events quick view

Input Set :	<b>generated-set-0</b>
Assertions failures :	<b>0</b>
Http failures :	<b>0</b>

#### Details on the events follow

##### USED INPUT-SET : GENERATED-SET-0

GET  
get https://us-east.apitest.apiconnect.ibmcloud.com/app/api/examples/retail/products

**Latency: 11ms**      **Download: 28ms**

**STATUS: 200**

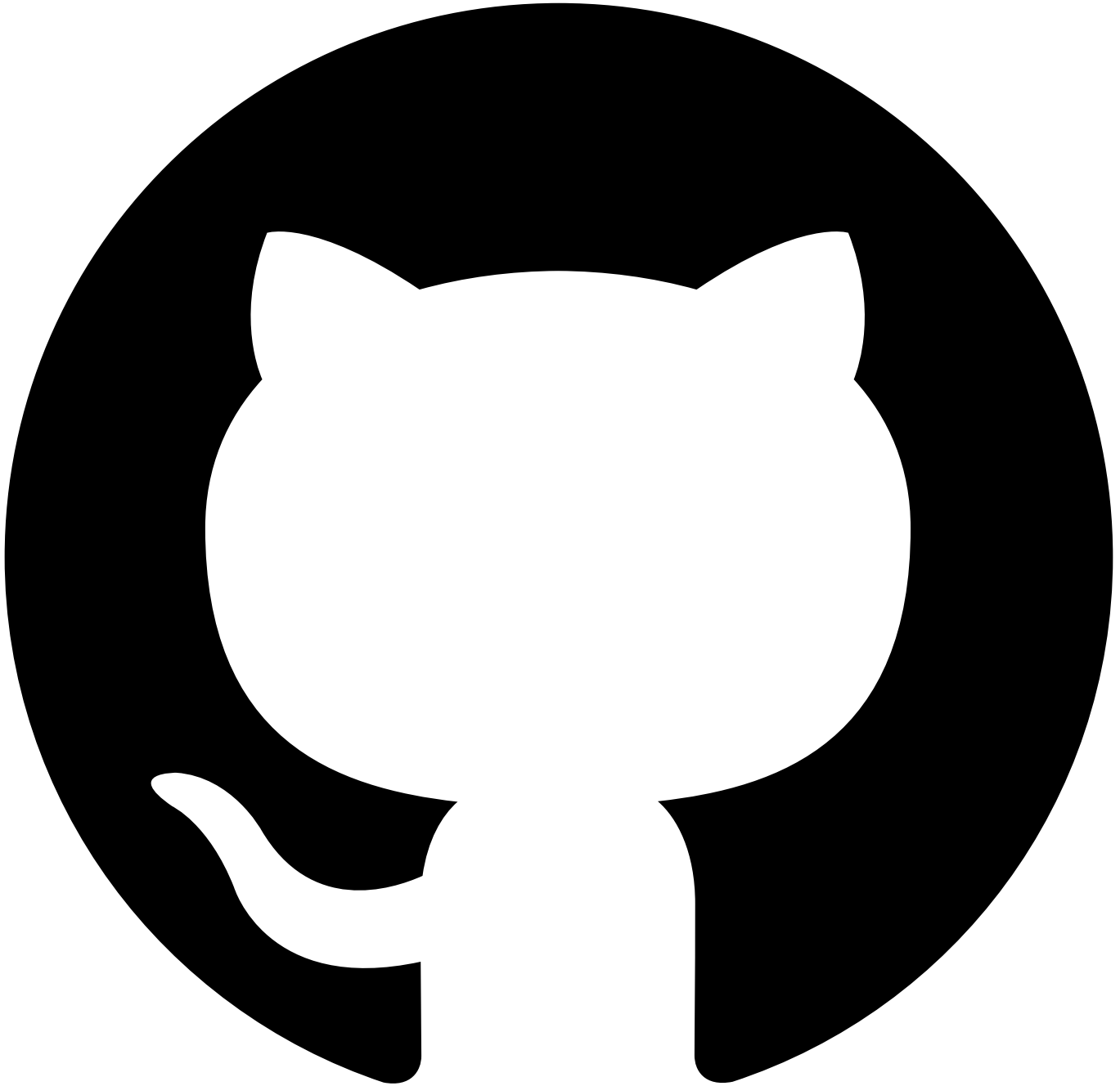
[See Request](#)

## Troubleshooting

If an Internal Server Error message is displayed during the test, follow the instructions in [Opening a test in the Editor](#) to return to the Editor page, and run the test again.

## What to do next

- Next: [Publishing a test](#)
- Previous: [Modifying a test](#)



Contribute in GitHub: [Edit online](#)

---

## Publishing a test

After you have completed the configuration of your test, and run it manually to confirm that it works correctly, you can publish the test.


Following on from publishing a test, you can continue to work on the test in the Composer without affecting the published test, and then publish the test again later to update the published test.

To publish a test, complete the following steps:

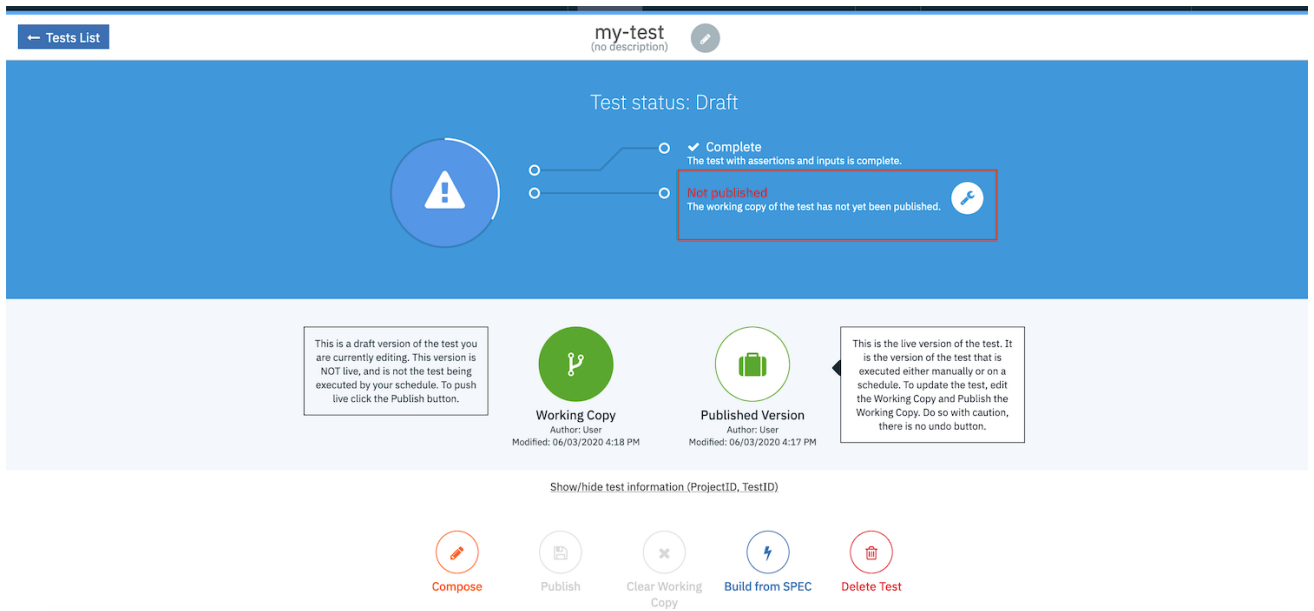
1. Click your test suite name in the menu bar to get to the "Test Suites" page.




2. On the "Test Suites" page, click the **Tests** icon [Tests](#) for the test suite that contains the test that you want to publish. The list of tests is displayed.

3. From the list of tests double-click the test that you want to publish, or click the **edit** icon  for the test.

4. The status page for your test is displayed. If your test has not been published before, the status will show as **Not published**.

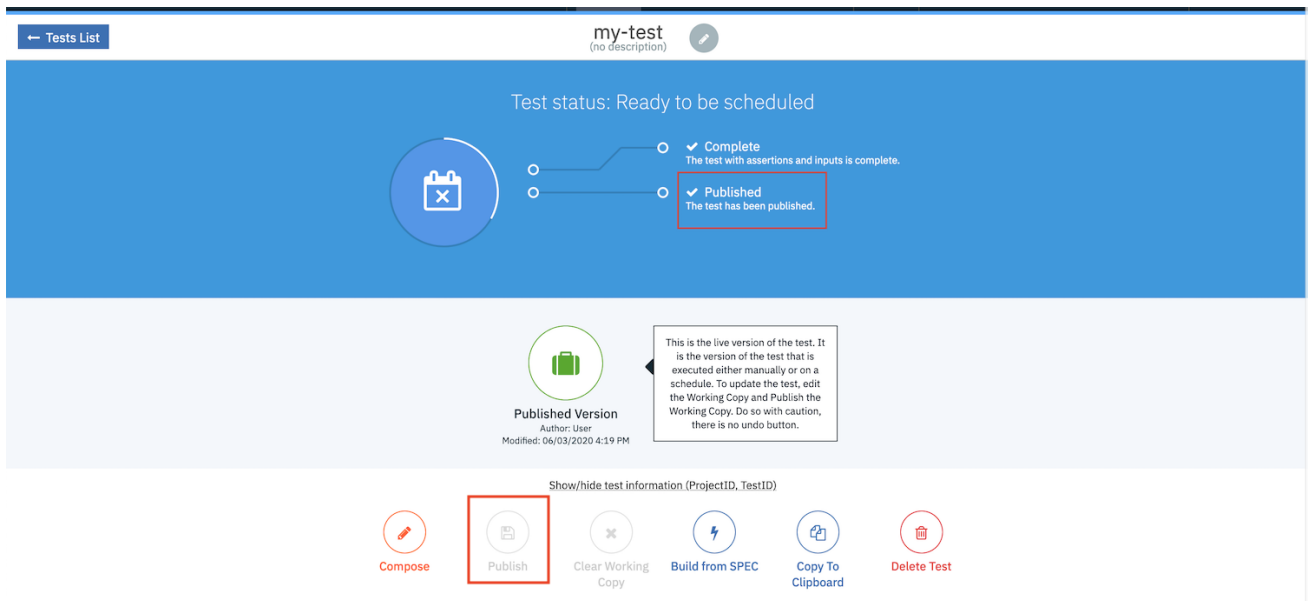


The screenshot shows the test status page for 'my-test' (no description). The test status is 'Draft'. A warning icon is visible. The status is 'Draft' and the test is 'Not published'. The working copy of the test has not yet been published. The test is currently in a 'Draft' state. Below the status, there are two versions: 'Working Copy' and 'Published Version'. The 'Working Copy' is the current version being edited. The 'Published Version' is the live version of the test. Below the versions, there are several action buttons: 'Compose', 'Publish', 'Clear Working Copy', 'Build from SPEC', and 'Delete Test'.

5. Click the **Publish** icon  to publish your test.

6. The test status page is updated with the following changes:

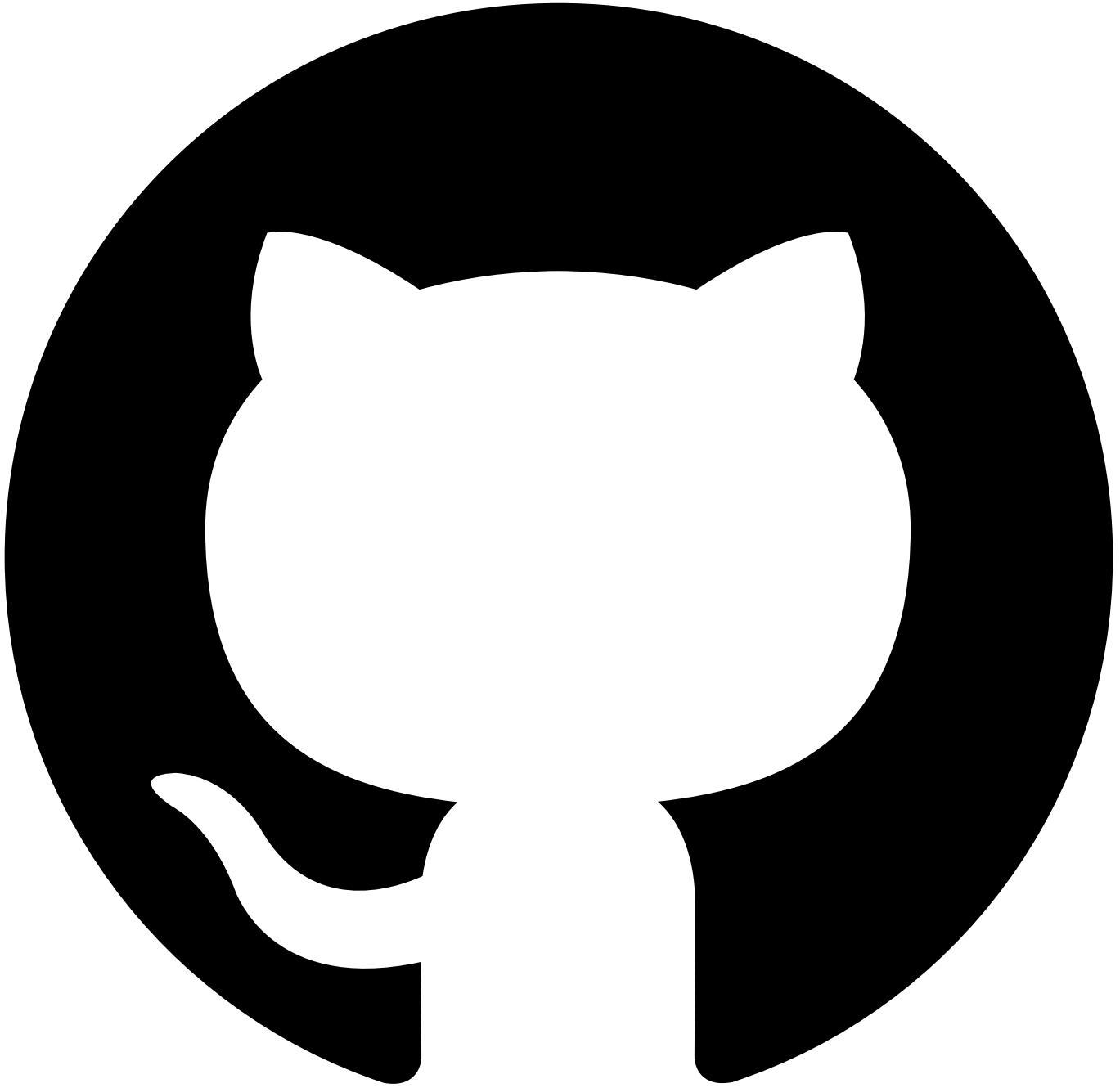
- The status of the test now shows as **Published**.
- The **Publish** icon is disabled.



The screenshot shows the test status page for 'my-test' (no description). The test status is 'Ready to be scheduled'. The test is 'Published'. The test has been published. The test is currently in a 'Published' state. Below the status, there is one version: 'Published Version'. Below the version, there are several action buttons: 'Compose', 'Publish', 'Clear Working Copy', 'Build from SPEC', 'Copy To Clipboard', and 'Delete Test'. The 'Publish' button is disabled.

## What to do next

- Next topic: [Tagging a test](#)
- Previous topic: [Running a test](#)



Contribute in GitHub: [Edit online](#)

---

## Tagging a Test

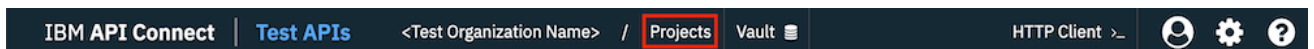
Adding tags to your tests is an easy way to keep them organized in your Test Suite. Furthermore, tags can be used as a way to selectively execute certain tests in your test suite when running them via an [API Hook](#) (see [Run Published Tests by Tag](#) for more information).


**Note:** To tag a test, you must have first [created a test](#).


---


## Adding a Tag to a Test

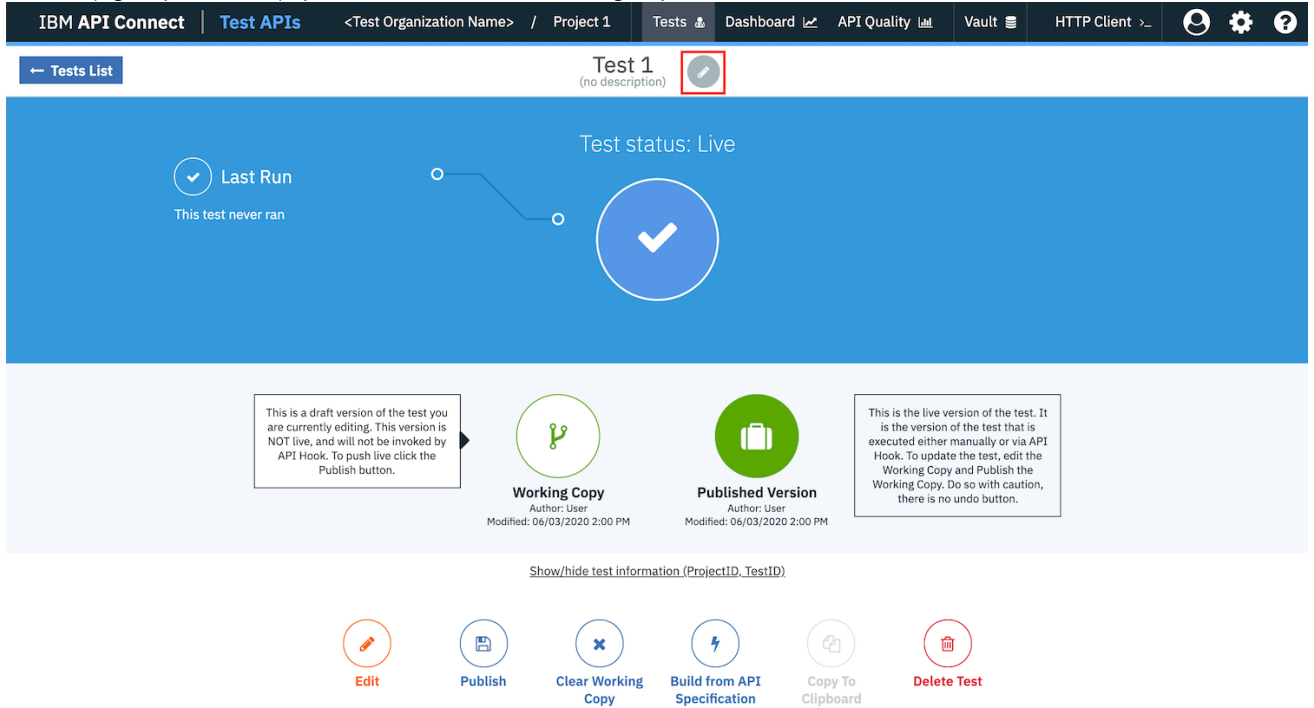
1. Click your test suite name in the menu bar to get to the "Test Suites" page.



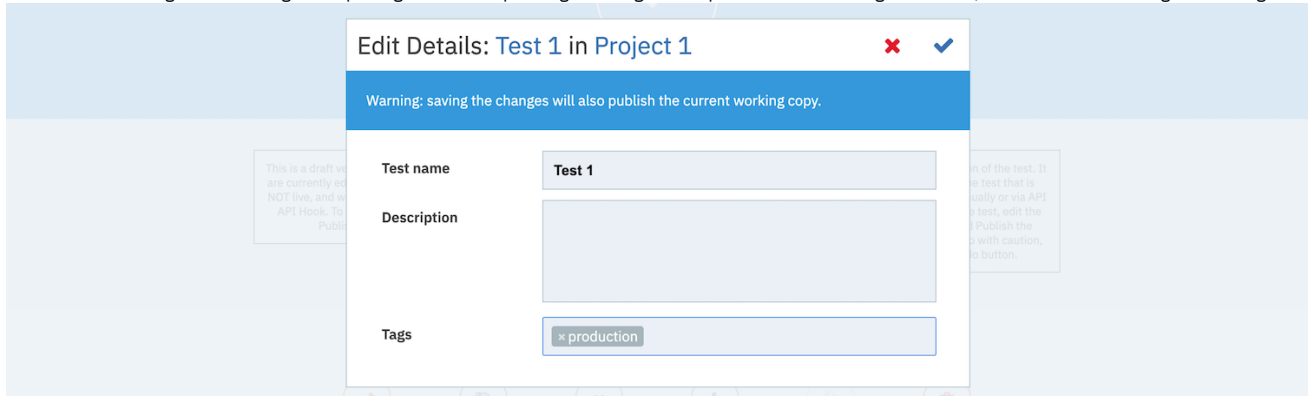
2. On the "Test Suites" page, click the **Tests** icon  for the test suite that contains the test that you want to tag. The list of tests is displayed.

3. From the list of tests double-click the test that you want to tag, or click the **edit** icon  for the test.

4. The status page for your test is displayed. Click on the **edit** icon  alongside your test name.

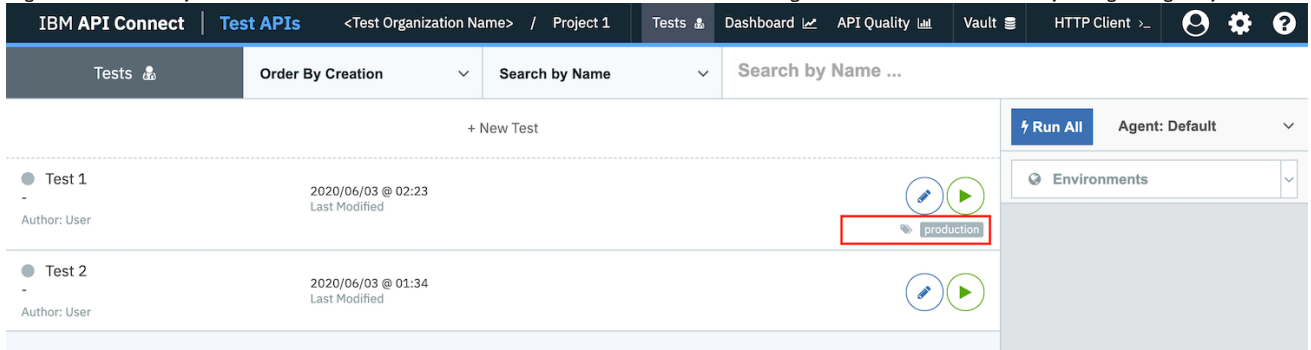


5. The Edit Details dialog box will open, allowing you to edit information about your test. To add Tags to your test, simply begin typing a name for a Tag into the Tags box within this dialog. You can assign multiple Tags to a test separating each Tag with a space. To remove a Tag from a test, click the **x** button alongside the Tag.



6. Click the **confirm** icon  to confirm and save your edits.

7. Tags can be viewed any time in the Tests List. Click the **Tests List** button  to go back to the Tests List and see your tags alongside your tests.

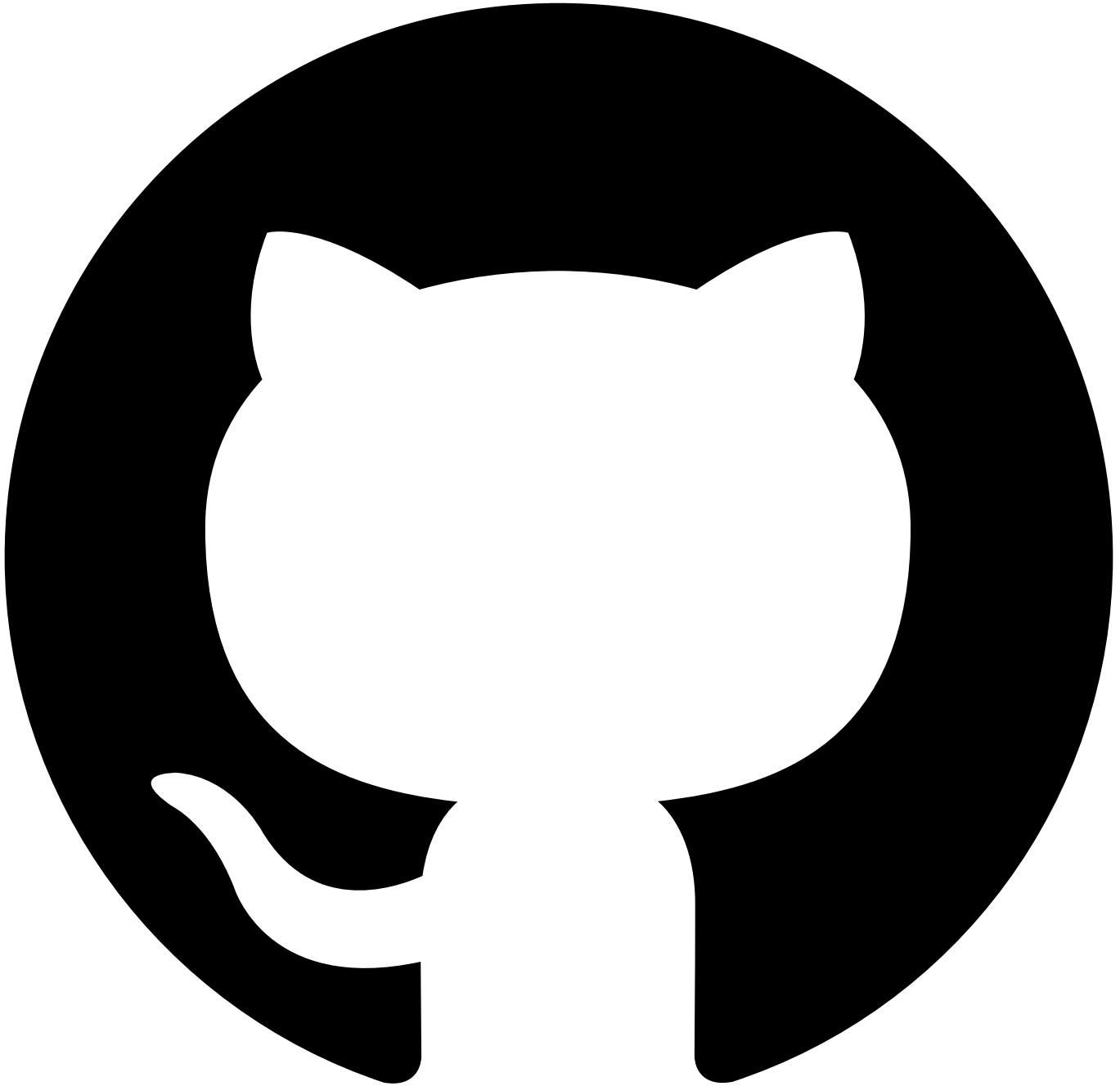


The screenshot displays the 'Tests List' interface in IBM API Connect. The top navigation bar includes 'IBM API Connect', 'Test APIs', and a breadcrumb path '<Test Organization Name> / Project 1'. The main content area shows a list of tests with columns for test name, author, and last modified date. A 'production' tag is visible on the right side of the test list. The right sidebar contains a 'Run All' button, 'Agent: Default', and an 'Environments' dropdown menu.

Test Name	Author	Last Modified	Tags
Test 1	User	2020/06/03 @ 02:23	production
Test 2	User	2020/06/03 @ 01:34	

## What to do next

- Next topic: [Create an API Hook](#)
- Previous topic: [Publishing a test](#)



Contribute in GitHub: [Edit online](#)

---

## Variables

Within API Testing there are multiple ways to define variables to address different requirements

### Defining variables

#### Global Variables

Global variables are ones that are common across the whole test and typically does not change between runs based on Input sets

#### Add a Global variable

To add a Global Variable within the Test Editor Select Data Sets Click **Add Global Variable** Provide Variable Name and value Click 

#### Edit A Global Variable



To edit an existing variable to change either the name or value For the given variable click

### Input Set Variables

The input set, is a group of input variables representing a scenario (e.g product id). Where the number of input sets determines how many times the test will be executed using those values.

If 3 input sets are defined, the test will run 3 times using the inputs defined within each of those sets on each run.

### Vault Variables

Variables can be defined within the vault which provides Test Suite scoped variables that can be used in all tests within that suite.

Vault variables can be defined as

- Variable - Visible
- Password - Hidden

### Hook Variables

When [Using API Hooks](#) it is possible to override Global defined variables within the request payload when running tests.

```
curl -X POST \
-H X-API-Key:{API-Key} \
-H X-API-Secret:{API-Secret} \
-H Content-Type:application/json \
-d \
 {
 "options": {
 "allAssertions": boolean,
 "JUnitFormat": boolean
 },
 "variables": {
 "url": "https://www.ibm.com",
 "username": "test",
 "password": "passw0rd"
 }
 }
 \
 \
<API-Hook-URL>/tests/{Test-ID}/run
```

### Variable Hierarchy

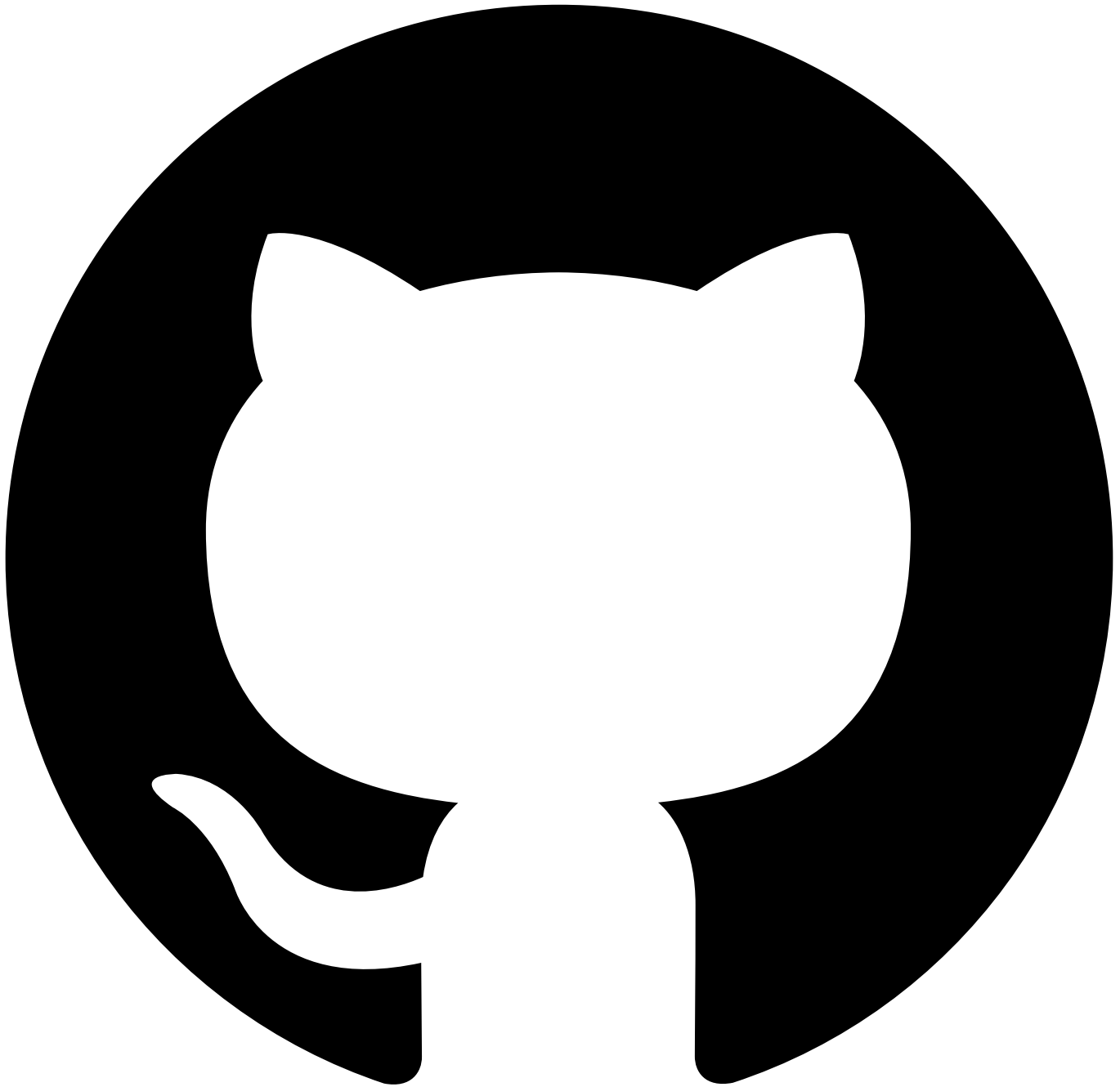
- If a variable is defined in Vault and either in Global or Input sets, Global or Input will take precedence
- If a variable is defined in both Global or Input set, Input set takes precedence
- If a variable is supplied as part of a Hook request and is defined in Vault, Global or Input set, the Hook will take precedence

## Referencing variables within a request

---

For setting variables during a test refer to [Set](#)





Contribute in GitHub: [Edit online](#)

---

## Insights-Driven Test Generation

**Available only as part of Cloud Pak for Integration 2021.2.1**


Insights-driven test generation takes the OpenTelemetry data from a source deployment and compares the observed behaviors to your current set of API Management test cases to find untested and non-obvious behaviors inherent within API calls in a specific environment.

- [Enabling Insights for Test Suites](#)
- [Generating Test Cases from Insights](#)
- [Watson Insights](#)
- [Troubleshooting](#)

The insight generation process is made up of the following stages:


1. Training and testing
2. Scoring
3. New behaviour generation
4. Test template generation


During insight generation you'll be able to see the current stage of the generation process on the insights panel:

Tests	0	
Events	0	
Failures	0	
Insights  Processing	?	

<b>Dashboard</b>	<b>Tests</b>	<b>API Quality</b>	<b>Settings</b>

Restart analysis 
×



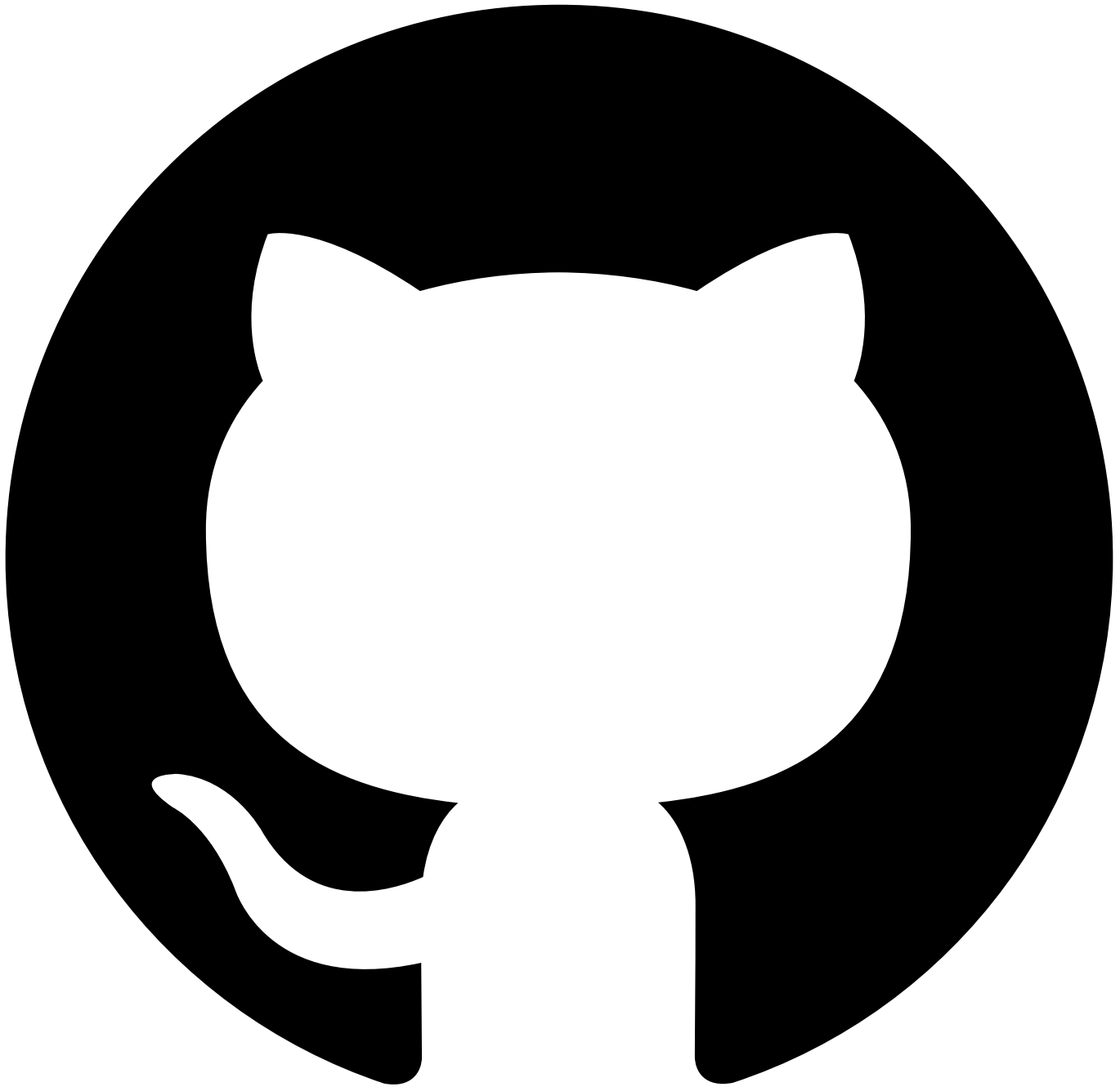
## Analyzing Data...

You can still create tests however their data might not contribute to insight generation.

- Training & testing
- Scoring
- Getting new behaviours
- Generating test template

### Known limitations

- API Management Spaces are not currently supported.
- The user currently has to manually refresh the tests page in order to obtain the most up-to-date insights status.



Contribute in GitHub: [Edit online](#)

---

## Enabling Insights for Test Suites

The following details are required in order to enable insights-driven test generation for a new or existing test suite:

### Jaeger

---

To configure the Jaeger connection details the following values are required:

*To obtain the service names used to identify the Jaeger traces from the API Gateway, review the options under the service dropdown on the Jaeger UI; the service name will be **apiconnect** by default.*

*The test service name is for the Jaeger instance deployed in the same environment as test APIs; source is the remote Jaeger instance.*

- **Service in test:** Test API Gateway Jaeger service name for the local system defined Jaeger instance
- **Service in production:** Source API Gateway Jaeger service name for the remote system defined in the Data service section
- **Time range:** Select the date range from which to obtain OpenTelemetry traces
- **Results limit:** Sets the number of OpenTelemetry traces that are to be retrieved from the source environment for use to build the training model

## Data Service Connection Details

---

Define the connection details of the remote data service to enable connection to the source environment to retrieve OpenTelemetry and analytics data:

- **Endpoint:** The route created for the insights data service
- **API key:** The key can be found in the `ai-data-service-api-key` secret mounted to the data service, this may need to be retrieved by your administrator
- **TLS certificate:** The certificate presented on the endpoint - if signed by an internal certificate authority or uses a self-signed certificate then please provide the full certificate

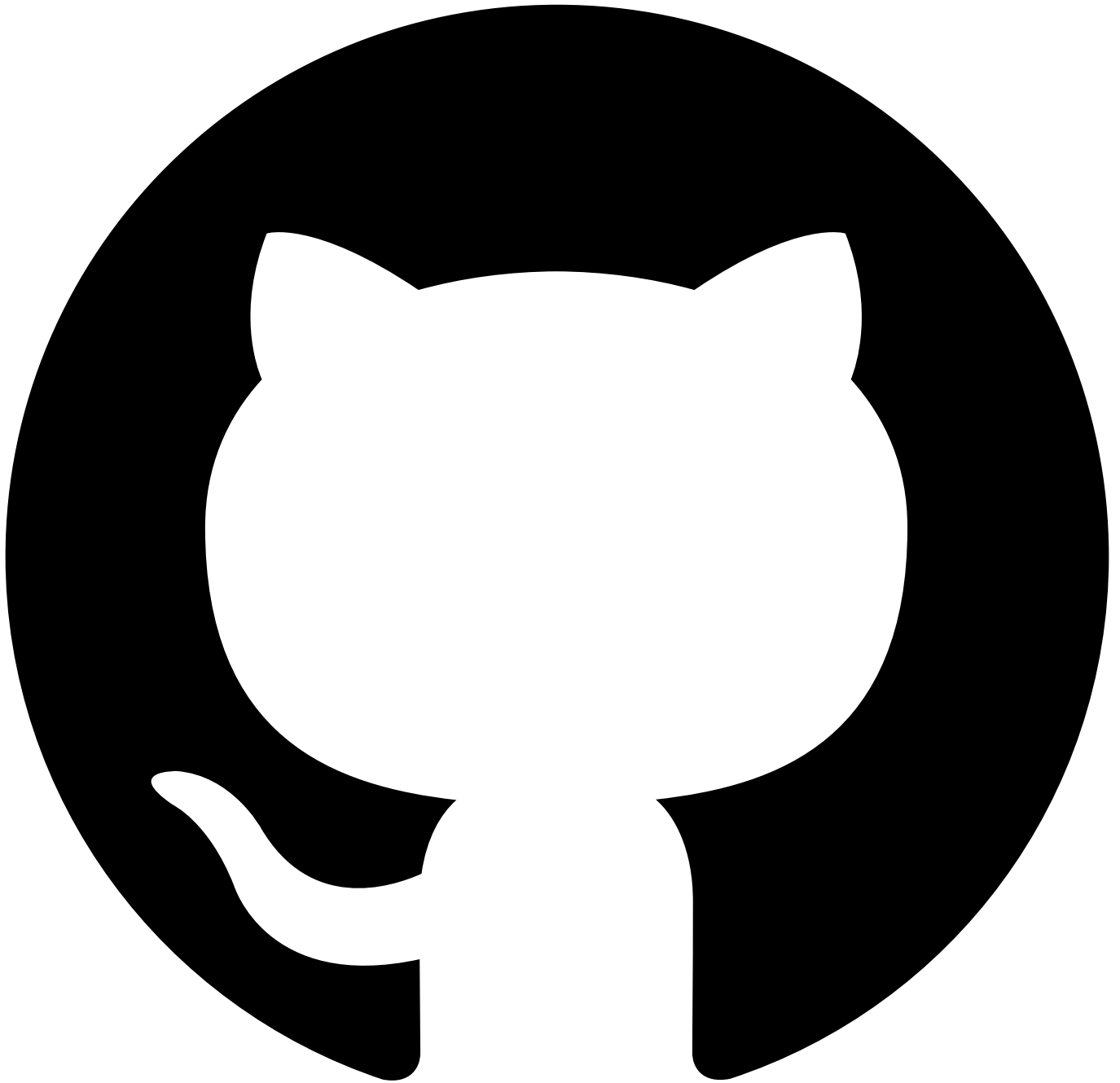
## API Management Connection Details

---

Provide the definitions for the source API Management deployment to enable the retrieval of analytics data.

- **Provider organization:** The name of your API Management organization
- **Catalog:** Name of the catalog in the above organization that you wish to use
- **Analytics service:** The name of the deployed analytics service that you wish to use
- **Username:** The username of the user created to retrieve analytics data from the source API Management instance
- **Password:** Password for the above user
- **Realm:** The user realm which the user authenticates against, should be in this format: `provider/idp`
- **OpenAPI document:** The OpenAPI spec document describing the set of APIs to be used for insights generation
- **Client ID:** The id of the application registered for use with the API Management REST API - No longer required as of CP4I 2021.3.1
- **Client secret:** The secret of the application registered for use with the API Management REST API - No longer required as of CP4I 2021.3.1

If a client application has not been created, then you may use a local user registry and create a new user for analytics.



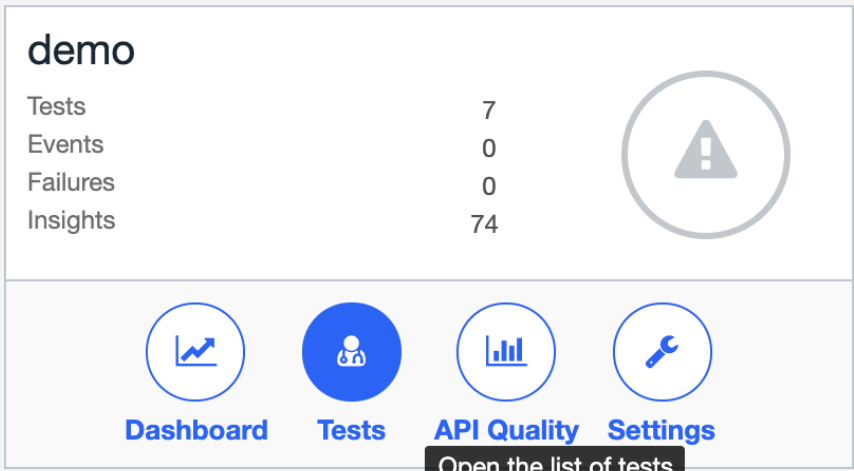
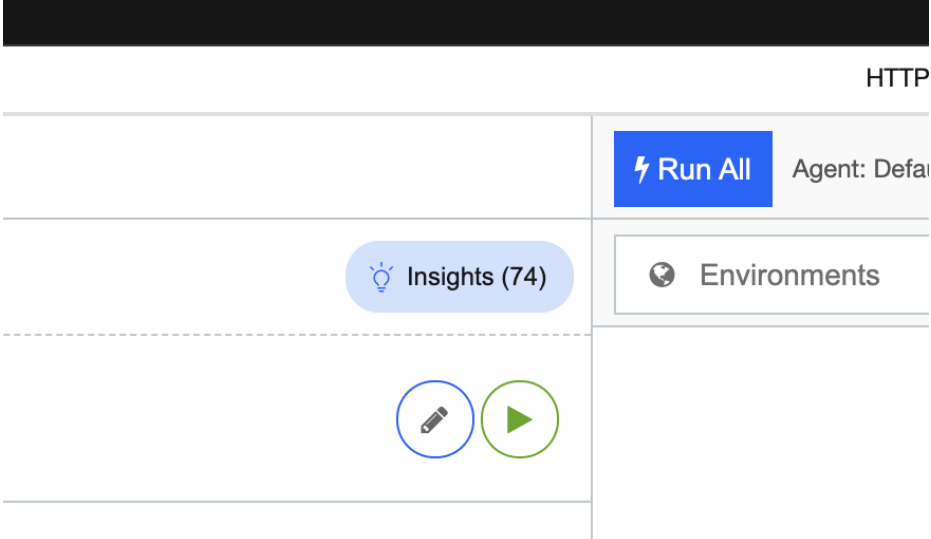
Contribute in GitHub: [Edit online](#)

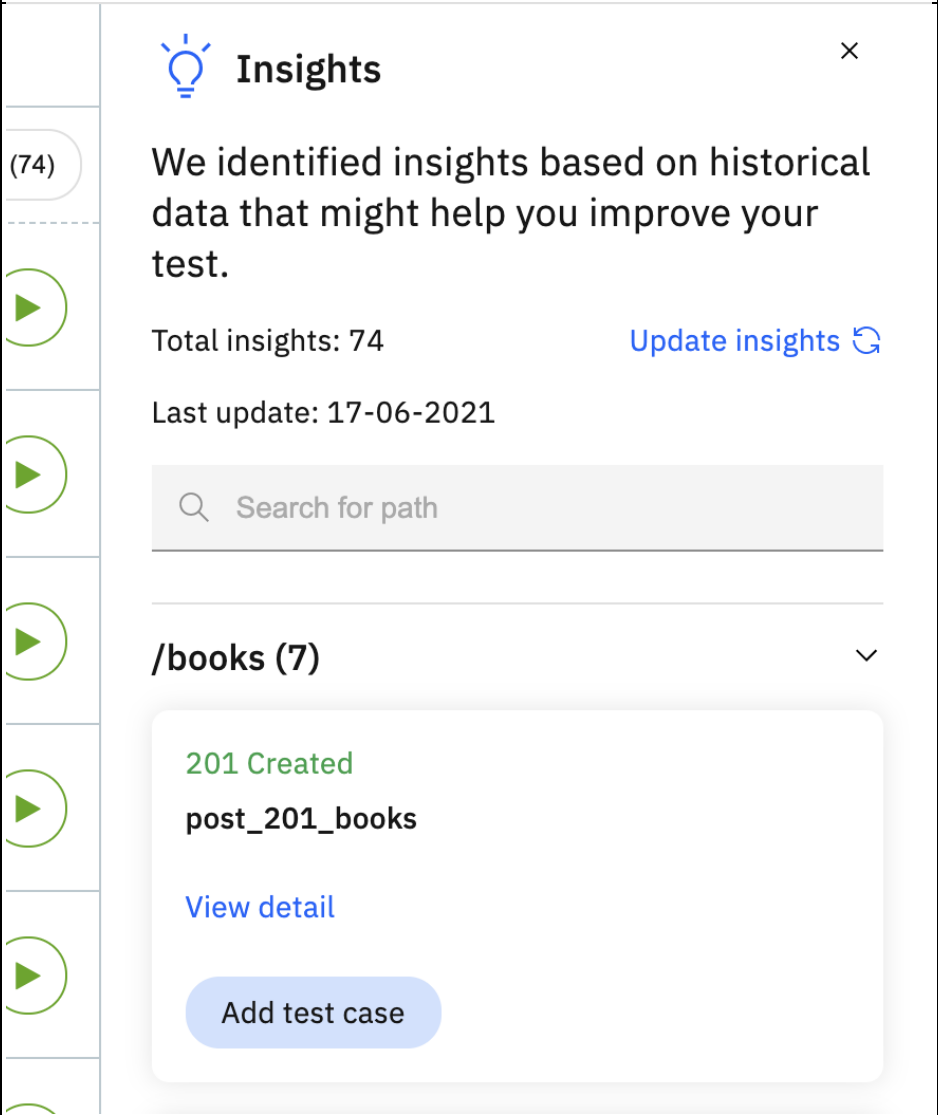
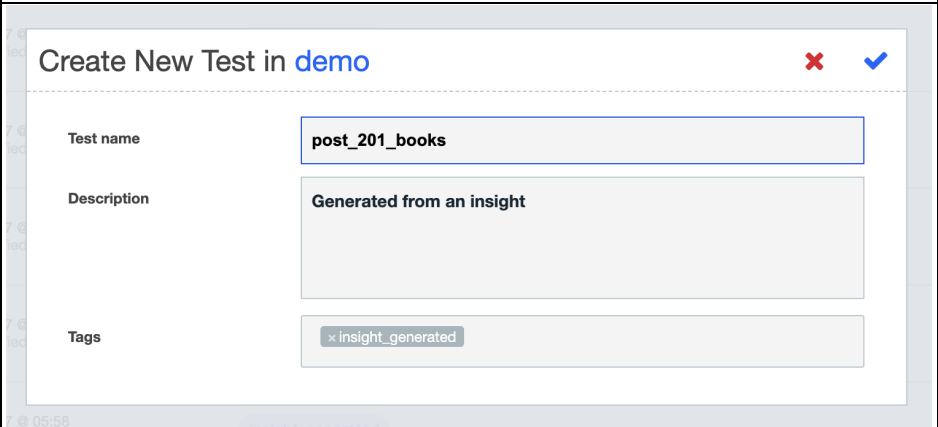
---

## Generating Test Cases from Insights

To make use of the Insights feature, create a new Test Suite with the details described in [Enabling Insights for Test Suites](#), and follow the steps below once the insights generation process has successfully completed:

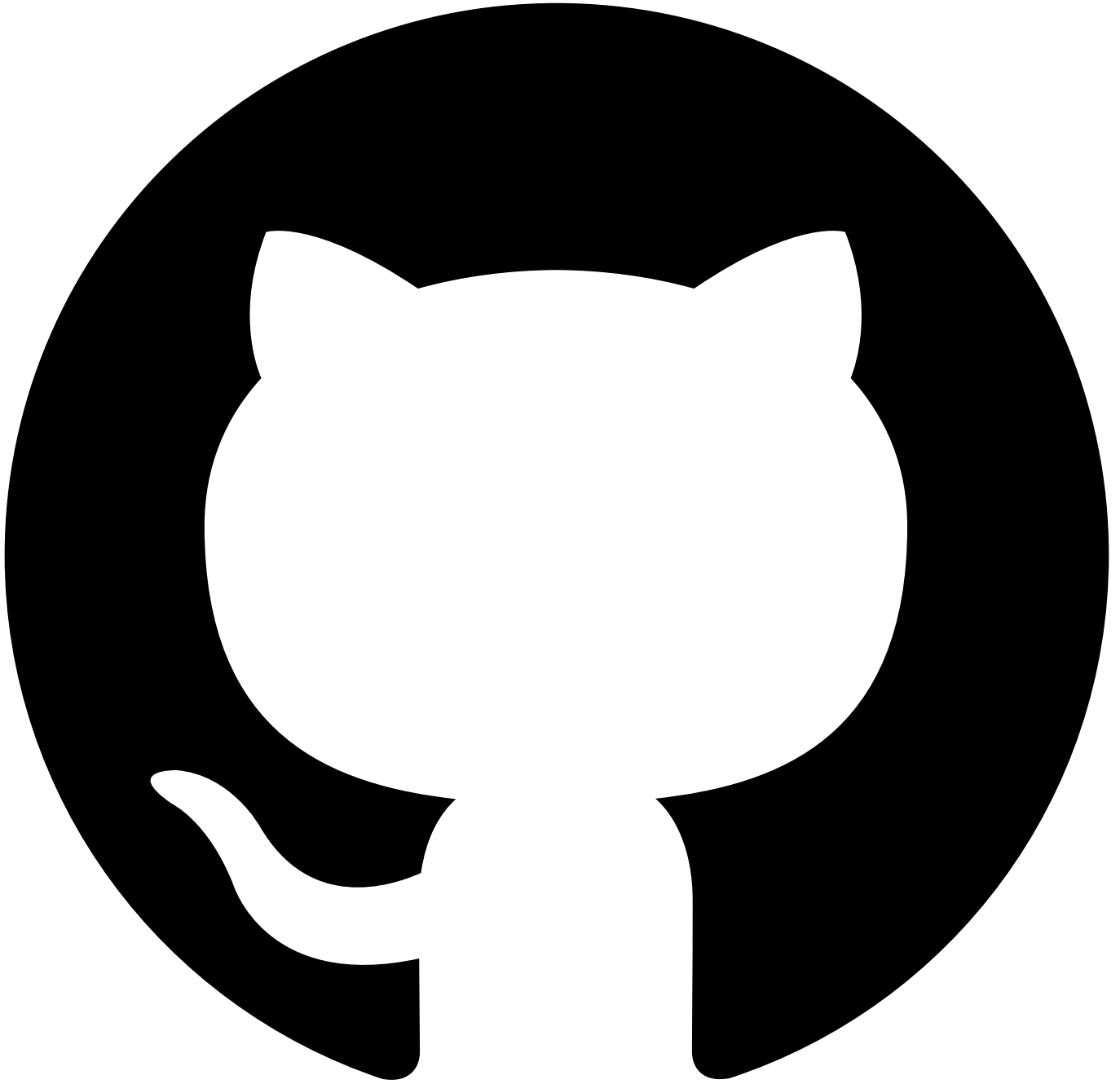
Step	Screenshot	Instruction
------	------------	-------------

Step	Screenshot	Instruction
1		<p>Click on the "Tests" button of the insights-enabled Test Suite to navigate to the tests dashboard</p>
2		<p>Click on the "Insights" button next to the "Create Test" button to bring the insights panel up</p>
3		<p>Click on the "Add test case" button on the desired insight</p>

Step	Screenshot	Instruction
		
4		Name the test case to be generated (and update its description and tags if desired) and click on the green tick to create the test case

Step	Screenshot	Instruction
5		<p>You should then edit the test case so that the <b>protocol</b>, <b>domain</b>, and <b>basePath</b> data set parameters match those of your deployment's</p>
6	<p>● <a href="#">post_201_books</a> </p> <p>Generated from an insight Author: admin admin</p> <p>2021/06/18 @ 02:40 Last Modified</p> <p><span style="background-color: #e0e0e0; border-radius: 10px; padding: 2px 5px;">insight_generated</span></p>	<p>The insight-generated test case should now show up on the tests dashboard (with the insights icon next to the test name)</p>





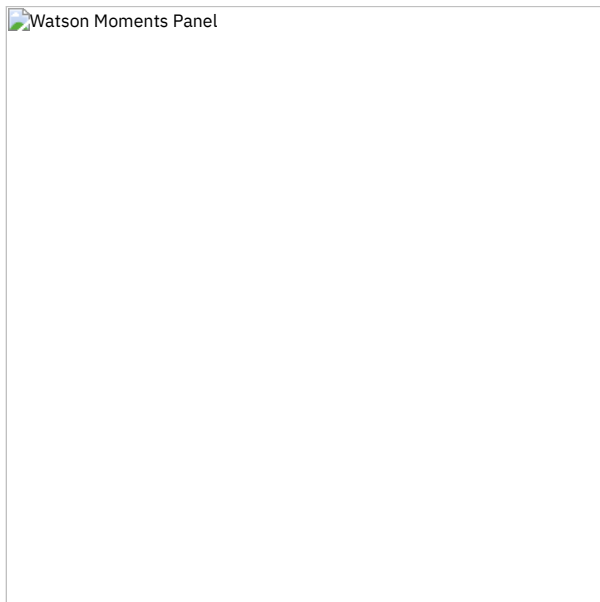
Contribute in GitHub: [Edit online](#)

---

## Watson Insights

**Available as of APIC release 10.0.4**

As an enhancement to the AI Test Generation feature introduced in the 2021.2.1 release of CP4I, AI generated insights now take the form of *Watson Insights*. Up to three test cases with an impact score of at least 70% can be recommended, as denoted by the **Impact score nn%** annotation on each tile.



Panel breakdown:

1 - Test coverage improvement

- By analyzing the amount of occurrences covered by each of the highlighted test cases we can calculate a number for the total potential coverage improvement for your API provided by utilising all of the recommended cases.

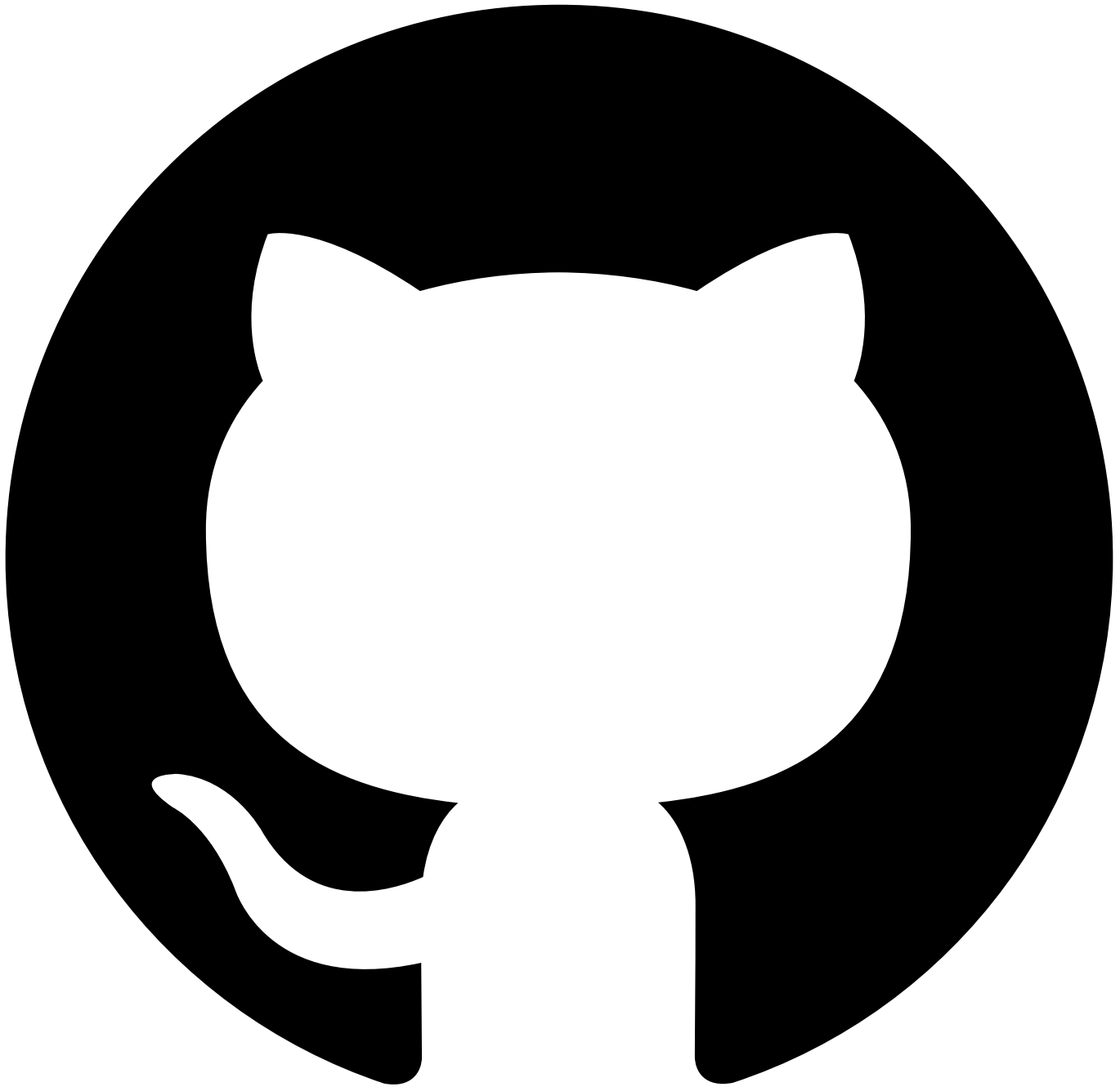
2 - Impact score

- An impact score should be seen as a quick "overview" of how impactful a test case would be to add. The score is calculated by comparing the overall usefulness of all cases and calculating a relative score expressed as a percentage. These values are shown in the tooltip seen in point 3.

3 - Impact score tooltip

- The values used to calculate the impact score are shown here. *Data completeness* describes how *complete* the test case is based on the values that are pre-populated for you. *Occurrences covered* describes how many traces from this set of results were covered by this single test case. The final *Impact Score* is calculated by taking these two values, comparing them with the other generated insights, normalising the data and combining them to calculate a geometric mean between the two values which creates our final score\*.

*\*This value may change in future updates as more data is used to further enhance AI Test Generation.*



Contribute in GitHub: [Edit online](#)

---

## AutoTest Assist

**Available only as part of the following API Connect editions:**

- **Cloud Pak for Integration 2022.2.1 and later (API Management capability)**
- **Enterprise as a Service (hosted on AWS)**

The AutoTest assistant (or, simply, the *auto-tester*) exercises your API by sending it a series of randomly-generated requests, guided by the API definition and any custom rules that you provide, and alerts you about any conformance errors and server-side errors that it observes.

You control the auto-tester by creating and running an AutoTest *profile*, and you can create multiple profiles against the same API to cover different use-cases. A profile specifies how the auto-tester should run, which endpoints it should test, and what kind of data it should send. A default profile is constructed so that you can get immediate feedback by providing no more than a target URL and credentials.

The AutoTest assistant is especially useful while an API is under development as it can quickly identify errors in an implementation without you having to write test cases. It is also valuable later in the cycle for spotting regressions, as the random nature of its requests can provide coverage that your existing test cases do not match. However, it is not meant as a replacement for other testing tools in API Connect, but is rather complementary, and the information it provides when it identifies an error is often a good starting point for creating a test case.

Follow these steps to create and run your first profile:

- [Creating a profile](#)
- [Running a profile](#)
- [Viewing the report](#)

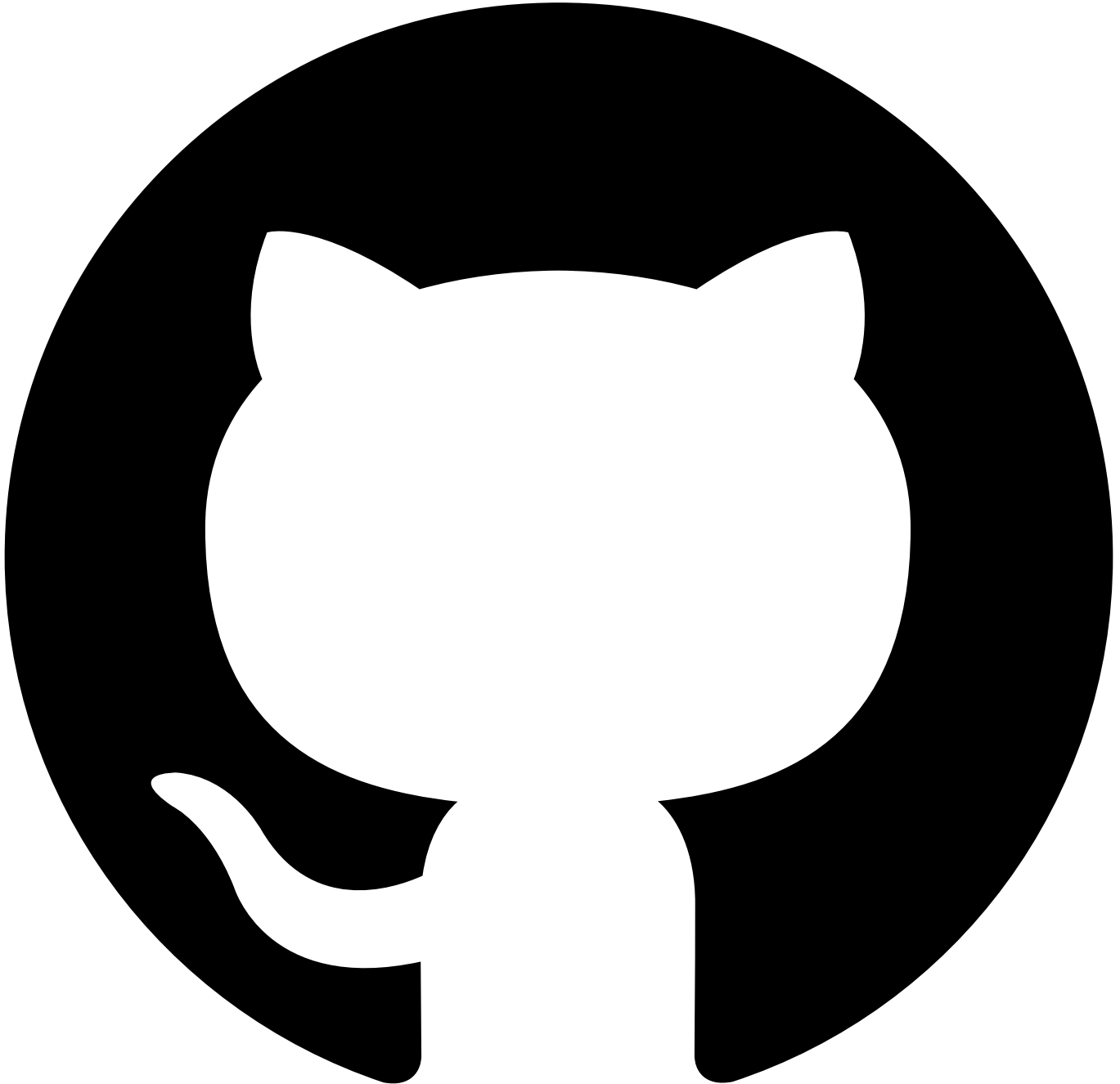
Consult these additional guides for detailed instructions on how to configure the auto-tester to support your use case:

- [Profile Configuration](#)
- [Profile Security](#)
- [API Extensions](#)
- [Data Generation Rules](#)

## Known limitations

---

- The API definition must conform to OpenAPI 2.0 (Swagger 2.0)
- The API definition document must have a `.json` or `.yaml` extension (you cannot select a document with another extension)
- If there are errors in the API definition then you will see only a blank screen, and the errors are not reported
- You must provide either a username and password, or a bearer token, to authenticate against the target API; other authentication methods are not supported
- You can run only one profile at a time

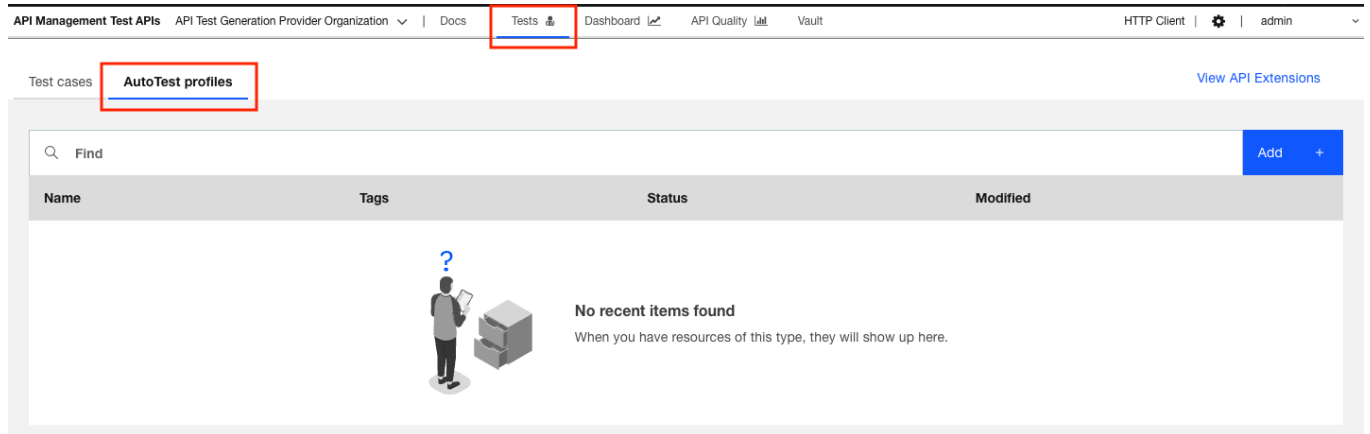


Contribute in GitHub: [Edit online](#)

# Creating an AutoTest profile

An AutoTest profile exists within a test suite, alongside your test cases.

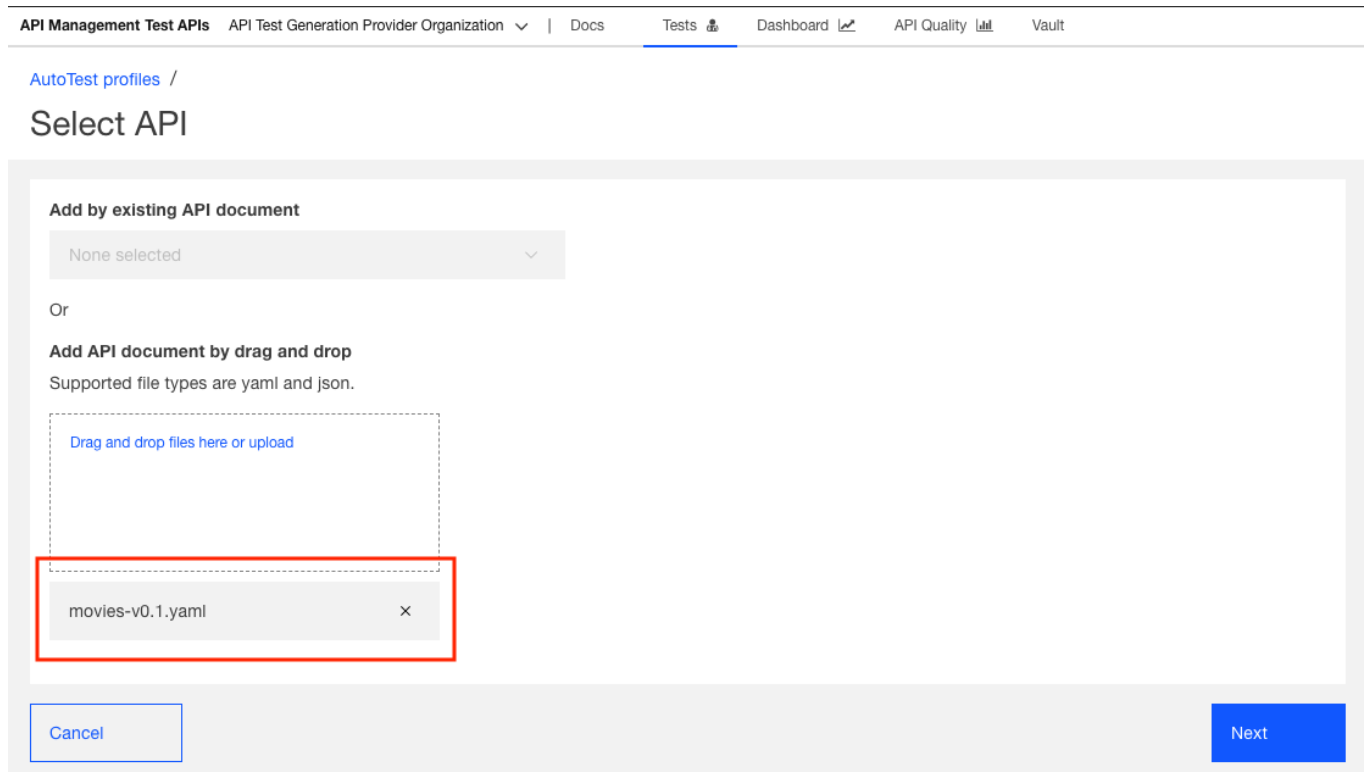
To add an AutoTest profile to a test suite: open the suite, select the **Tests** tab, and click on **AutoTest** profiles.



Click **Add**, and select the API you wish to target: choose either an existing API from the dropdown (from a definition you have uploaded previously) or upload a new definition by dragging and dropping an OpenAPI document in either JSON or YAML format. It doesn't matter if you upload the same document more than once as it will be recognized.

### Known limitations

- The API definition must conform to OpenAPI 2.0 (Swagger 2.0)
- The API definition document must have a `.json` or `.yaml` extension



Click **Next** and provide at least a name for your new profile. You can also add comments and tags.

[AutoTest profiles](#) /

## Profile info

### Info

---

Name

Description

Tags

This is a list of comma separated values

[Cancel](#) [Back](#) [Next](#)

Click **Next** and AutoTest scans the API document and extracts semantic information that will inform its generation of requests. You get a chance to review the information here, but if you need to change anything you must wait until the profile is complete. If this is your first time with AutoTest you can take the content on trust and move on, but read through [API Extensions](#) if you want to understand the detail.

### Known limitations

- If you see a blank screen at this point then there is most likely an error in your API definition, and you should validate it in the Swagger editor

[AutoTest profiles](#) /

## Review API Extension

**Info:** AutoTest has generated these extensions for the API you selected. You can edit the extensions once you have completed the profile.

```
1 properties:
2 - items:
3 - name: order_id
4 semantic: id
5 json_ptr: '#/definitions/Order'
6 - items:
7 - name: customer_id
8 semantic: id
9 json_ptr: '#/definitions/Customer'
10 - items:
11 - name: name
12 semantic: name
13 json_ptr: '#/definitions/ErrorTarget'
14 - items:
15 - name: message
16 semantic: sentence
17 json_ptr: '#/definitions/ErrorItem'
18 - items:
19 - name: phone
20 semantic: phone_number
21 - name: last_name
22 semantic: last_name
23 - name: first_name
24 semantic: first_name
25 - name: email
26 semantic: email
27 - name: username
28 semantic: username
```

Click **Create** and AutoTest will have created a profile with the default configuration.

[AutoTest profiles](#) /

## Summary

- ✔ Created AutoTest profile: movies - default.
- ✔ Generated associated API Extension.

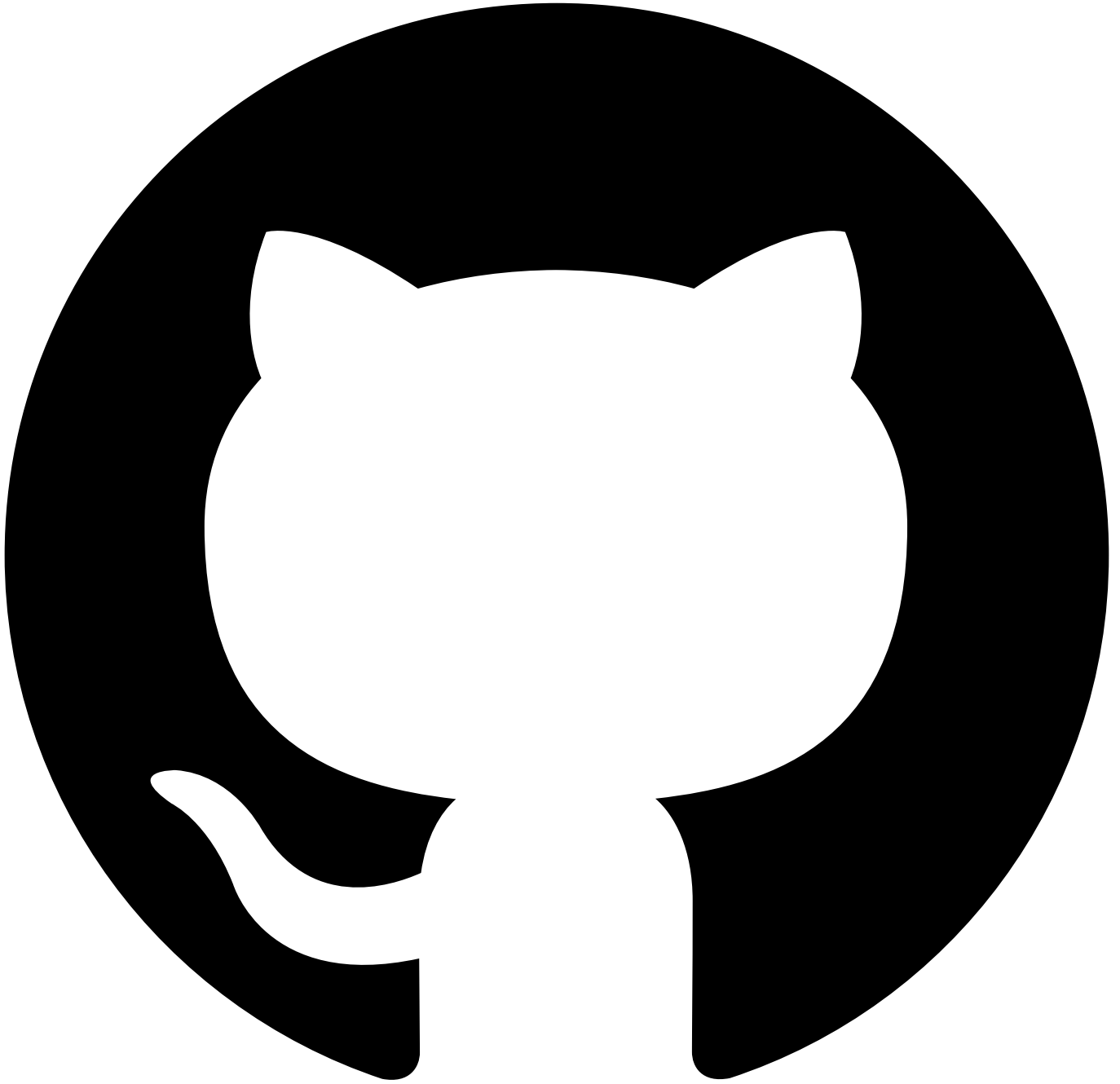
Click **Close** and it takes you back to the starting page, but now showing your new profile.

Test cases **AutoTest profiles** [View API Extensions](#)

Name	Tags	Status	Modified
movies - default	dev	Never run	3 minutes ago

Items per page: 10 | 1-1 of 1 items | 1 of 1 page

Alternatively, click **Edit** and it takes you straight on to the next step, which is [Running a profile](#).



Contribute in GitHub: [Edit online](#)

---

## Running an AutoTest profile

To be able to run an AutoTest profile you will need to configure it first. Make sure to set the ServerURL parameter correctly otherwise all the generated API calls will fail. For more information please refer to [AutoTest Profile Configuration](#). At any point during this process you can click on the save button to maintain your progress.

Once you have made your changes in the configuration panel, go to the security panel to set your authentication.



**Design**

**Report**

Configuration

Info

Security

For more information read through [AutoTest Profile Security](#).

Once you have made all your changes to the configuration and security. You can either save, or click on the run button.


This will take you to the report page where the exerciser will generate calls to test the API.

**Design**

**Report**

Overview

API calls

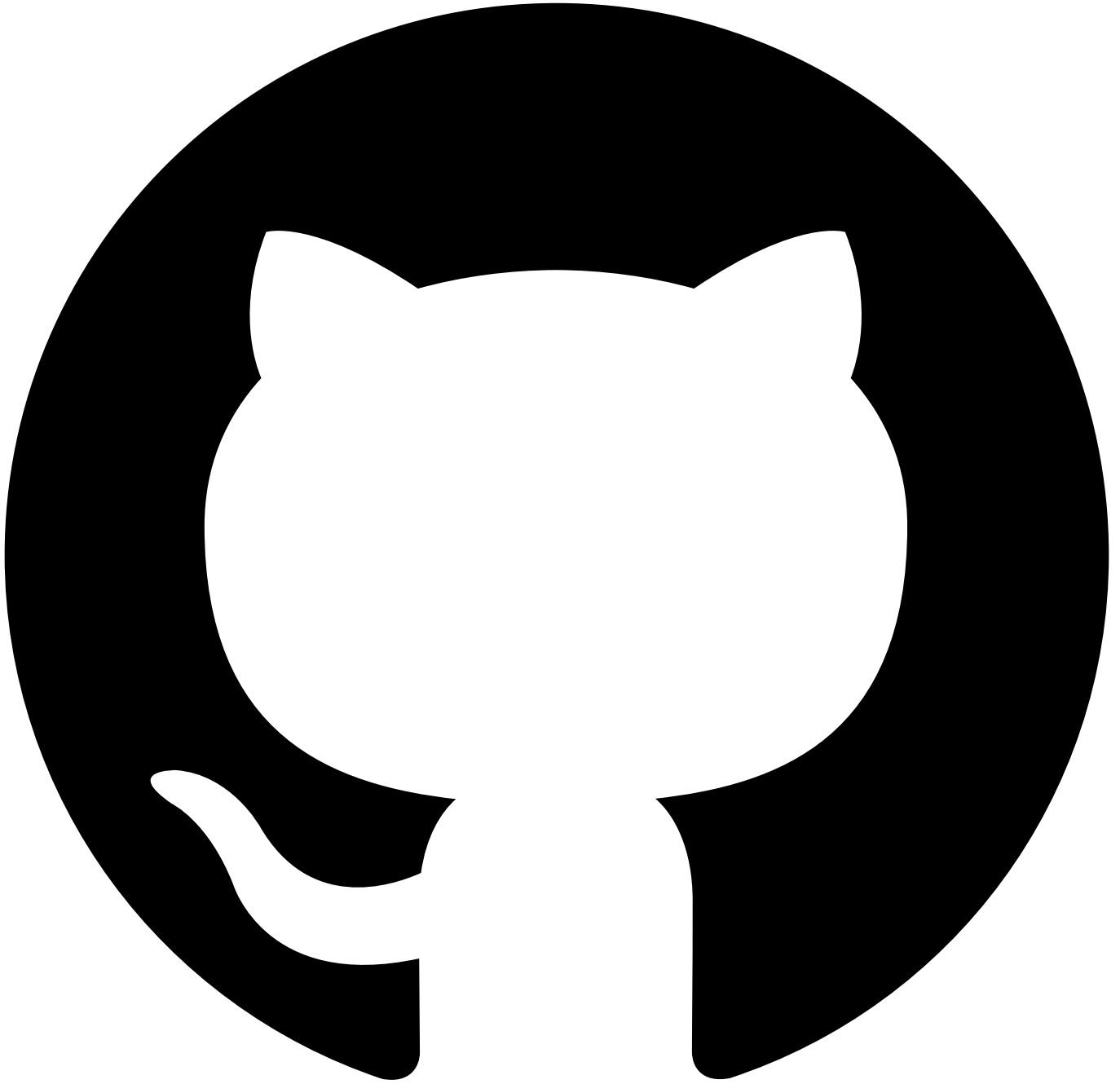
 Generating calls

Calls made so far: 181

Number of failures: 170

Stop

Once the generation has finished, it will take you to the next step [AutoTest Report](#).



Contribute in GitHub: [Edit online](#)

---

## Viewing the AutoTest report

Once the AutoTest run has completed (either stopped manually or the configured time to run threshold has been reached) you will be presented with the **Report** section, where you will be able to view the results of the run.

First up is the **Overview** tab, which contains a table displaying all the paths defined in the AutoTest profile's API specification, alongside their associated HTTP methods (verbs), and followed by the number of each class of responses received throughout the course of the AutoTest profile run.

AutoTest profiles /

# ACME Bookshop

Save Run

Design Report

Overview API calls

Last ran - a few seconds ago - | Error only  Off

Find							
Path	Verb	2xx successes	4xx successes	2xx errors	4xx errors	5xx errors	Total
/books <span style="color: red;">❌</span>	POST	0	0	0	307	0	307
/customers <span style="color: red;">❌</span>	POST	0	0	0	312	0	312
/services/author <span style="color: red;">❌</span>	POST	0	0	0	911	0	911
/services/category <span style="color: red;">❌</span>	POST	0	0	0	886	0	886
/services/usage <span style="color: red;">❌</span>	POST	0	0	0	955	0	955

Items per page: 10 | 1-5 of 5 items | 1 of 1 page

The **API calls** tab contains a more detailed view of every API call made by the run. The left side contains the list of the calls and their respective response statuses; the search bar lets you filter on the HTTP method, the path, and the response code; this should make it relatively easy to find specific requests amongst the potentially large number of API calls.

You can also get to the details view by clicking on any of the non-zero numbers in the overview table; this will automatically filter the list of API calls by the appropriate class of responses (or just the path and verb if the total tally is clicked on), as shown below.

AutoTest profiles /

# ACME Bookshop

Save Run

Design Report

Overview API calls

Last ran - a few seconds ago - | Error only  Off

Q	POST /books 4	x
Total calls (307)		
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖
POST /books	404	✖



### No call selected

First select a call from the left side navigation and its content will show up here.

Either of the two views lets you apply a global **Errors only** filter by interacting with the toggle in the top right corner, under the **Save** and **Run** buttons. This will apply a filter to both the overview table and the detailed API calls list so that they will either display only errored or all requests.

Selecting one of the calls from the list will render the details of that call on the right hand side. As seen below information such as headers and bodies for both the request and response is shown. This lets you closely examine precisely how a specific API endpoint had been called and the response the request received.

POST /books 4

Total calls (307)

- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404
- POST /books 404

# POST/books 404

## Error

Illegal returnStatus code 404

## Request

URL: https://ademo-gw-gateway-atm.apps.fermi-gyges.cp.fyre.ibm.com/books

Verb: POST

- Headers:**
- Authorization: Bearer \*\*\*\*\*
  - Content-Type: application/json

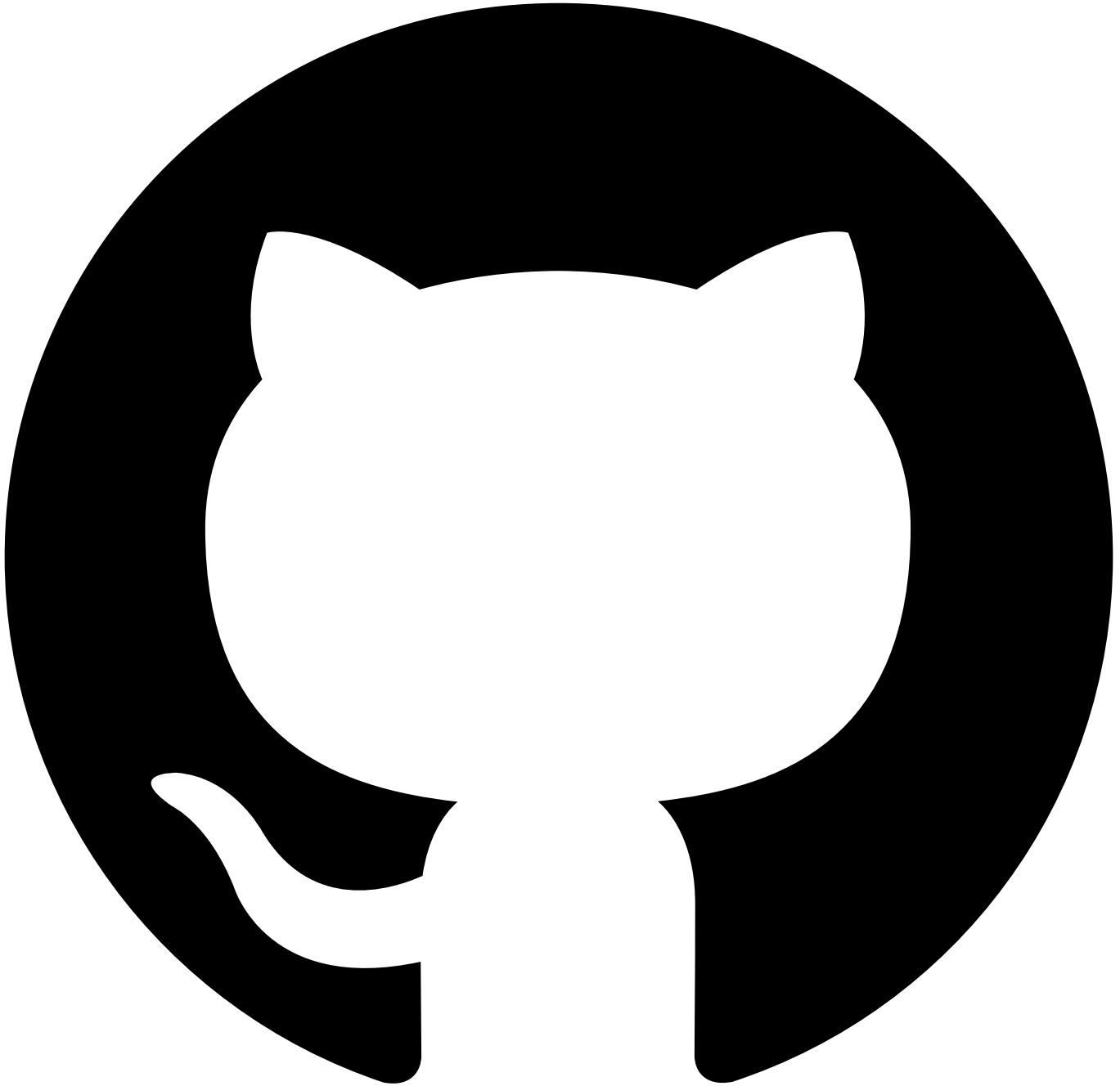
**Body:** {  
 "author": "Marta Monahan",  
 "date": "1943-12-31",  
 "format": "paperback",  
 "isbn": "VyGkneaKhbUOBF0",  
 "language": "Welsh",  
 "publisher": "qfD3Eq1fk",  
 "title": "W9Nx4Jwx47mKRaibUYcg4HvH-ra9"  
 }

## Response

Status code: 404

- Headers:**
- Content-Type: application/json
  - Date: Tue, 28 Jun 2022 13:06:55 GMT
  - Retry-Count: 0
  - Uber-Trace-Id: 2313f6af8e5d40dd:2313f6af8e5d40dd:0000000000000000
  - Uberctx-Appliance: ademo-gw-0
  - Uberctx-Clientip: 10.254.16.1
  - Uberctx-Domain: apiconnect
  - Uberctx-ErrorMessage: The%20requested%20URL%20was%20not%20found%20on%20this%20server
  - Uberctx-Errorreasonphrase: Not%20Found
  - Uberctx-Gtidinternal: c6a9902a62bafcef0188b380
  - Uberctx-Httprequesturi: %2Fnull
  - Uberctx-Httprequesturl: null
  - Uberctx-Httpversion: HTTP%2F1.1
  - Uberctx-Iserror: true
  - Uberctx-ObjectName: apiconnect
  - Uberctx-Objecttype: apigw
  - Uberctx-Responsesize: 111
  - Uberctx-Tid: 25736064
  - Uberctx-Tmz: %2B00%3A00

**Body:** {  
 "statusCode": "404",  
 "httpMessage": "Not Found",  
 "moreInformation": "The requested URL was not found on this server"  
 }



Contribute in GitHub: [Edit online](#)

---

## Profile Configuration

The configuration controls the execution of your profile; it determines for how long the test should run, and what kinds of requests it should generate. A newly-created profile includes a default configuration that you can run immediately to check that the auto-tester works with your API. The one thing you may have to set or change before you can run is the target [Server URL](#), which must point to your test deployment; the default value is copied from the API definition, where one is provided. You must also specify appropriate authentication settings on the separate **Security** tab (see [Profile Security](#)).

The configuration is in YAML format. The editor highlights syntactic errors as you type, and verifies the content for consistency with the API definition each time you press **Save**. If there is an error in the configuration, then the profile cannot run.

## The Profile Configuration editor

---

## Configuration

Info

Security

```

1
2 #####
3 # Default profile for the API Exerciser
4 #####
5
6
7 # The API
8 #####
9
10 # The endpoint to test (must be set)
11 ServerURL: 'https://api-host:8000/apis'
12
13
14 # Stopping Criteria:
15 # the tester will stop when the first of these limits is reached
16 #####
17
18 # The maximum length of time to run, in minutes
19 TimeToRun: 2
20
21 # The maximum number of requests to send (0 = unlimited)
22 MaxRequests: 0
23
24 # The maximum number of errors to accept (0 = unlimited)
25 MaxErrors: 3
26
27
28 # Execution Control
29 #####
30
31 # Whether to submit requests in parallel:
32 # this can be helpful in exposing concurrency issues in the API
33 ParallelRequests: true
34
35 # Whether to submit known, badly-formed requests
36 ErrorInjection: false
37
38 # What percentage of requests should have errors
39 # when ErrorInjection is true
40 ErrorPercent: 1.5
41

```

Sections in the configuration:

- [The API](#)
- [Stopping Criteria](#)
- [Execution Control](#)
- [Resources](#)
- [Operations](#)
- [Datatypes](#)

## The API

### ServerURL

The server URL is the single endpoint to which all requests are sent. It must be a valid URL, reachable from your namespace, and include at least a scheme and a server address; it may also include a port number and a path prefix. It must not include a trailing slash. The server URL is prefixed to an operation path from the API definition to create the target of a request. For example, given this URL definition from the configuration:

**ServerURL:** `https://api-host:8000/apis`

And this operation path from the API definition:

```
paths:
 /movies:
 post:
```

The auto-tester will send `POST` requests to this target URL:

`https://api-host:8000/apis/movies`

## Stopping Criteria

The profile stops running as soon as any of these criteria are met.

### TimeToRun

The maximum length of time to run, in minutes: the auto-tester stops sending requests once this time period has elapsed. The overall running time will be longer because it includes additional time for setup and teardown.

For example, to set a maximum running time of ten minutes:

**TimeToRun:** `10`

## MaxRequests

The maximum number of requests to send: the auto-tester stops issuing requests once this number has been reached. The final total of requests may be greater if additional requests are needed for setup and teardown.

Set this to zero to disable the request limit, and then the auto-tester will issue as many requests as fit into the allotted time period.

For example, to set a maximum of two thousand requests:

```
MaxRequests: 2000
```

## MaxErrors

The maximum number of error responses to allow: the auto-tester stops issuing requests once this number has been reached.

An error in this context is not an API error indicated by a **4xx** (or even, sometimes, a **5xx**) response, which is to be expected if the profile exercises all execution paths through the implementation. An error here is one where the response to a request is inconsistent with the API definition, which is a defect. Common cases include:

- the status code is not documented
- the payload does not conform to the schema
- any 500 error

There should be no such errors in a mature implementation so the limit is typically set low, so that the profile can terminate early if they occur. You can set the limit to zero and disable this stopping condition, which might be appropriate for a development environment where you know that some operations have defects (or you can disable those operations from ever being called as described under [Operations](#)).

For example, to set the error limit to three:

```
MaxErrors: 3
```

## Execution Control

---

### ParallelRequests

If this is **true** then the auto-tester spawns threads to issue multiple requests concurrently, otherwise it issues just one request at a time. The concurrent option increases throughput and can be helpful in exposing synchronization issues in the API.

The number of concurrent threads is fixed and cannot be changed.

For example, to disable concurrent requests:

```
ParallelRequests: false
```

### ErrorInjection

If this is **true** then the auto-tester issues a certain percentage of malformed requests to further exercise the validation logic of the API, otherwise it only issues requests that are well-formed according to the API definition.

The percentage of requests is specified as **ErrorPercent**.

For example, to enable malformed requests:

```
ErrorInjection: true
```

### ErrorPercent

The percentage of malformed requests, when **ErrorInjection** is **true**.

For example, to include 2% of malformed requests:

```
ErrorPercent: 2.0
```

## Resources

---

The AutoTest execution model is based around the notion of "resources", where a resource has a schema (of object type) and a set of operations (or methods). Resources are defined in the [API extensions](#).

The auto-tester only exercises operations that belong to resources. To decide which operation to execute next, the auto-tester first picks a resource type at random, and then an operation from that type (again, at random).

You control the frequency with which resources are selected by assigning them weights. The selection frequency for a given resource is computed as the weight for that resource divided by the sum of all the weights.

A zero weight means that the resource will never be selected, and you can use this to exclude certain resources from testing; for example, when you know that they are not yet fully implemented. It is an error for all weights to be zero which would leave nothing to test.

Use the **Resources** key to assign weights to specific resources, and **ResourceDefault** to set the default weight for any resources not specified.

### Resources

The weights assigned to specific resource types. The value is an object, where the keys are resource names, and the values are non-negative integer weights. Resource names must be declared in the API extensions.

For example, to increase the test focus on **Customer** operations, and exclude **Order** operations entirely:

```
Resources:
 Customer: 2
 Order: 0
```



## ResourceDefault

The default weight for any resource type not mentioned explicitly under **Resources**. The weight can be any non-negative integer value, but there are two common use cases:

- if the weight is 1 then all resources are selected with equal frequency, except where specified otherwise under **Resources**
- if the weight is 0 then all resources are disabled, except for those specified otherwise under **Resources**

For example, to disable all resources by default:

```
ResourceDefault: 0
```

In this case there must be at least one non-zero weight under **Resources**.

## Instances

Upper and lower bounds on the number of resource instances that should exist at any time. The value is an object, where the keys are resource names, and the values specify lower and upper bounds for those resource types. Resource names must be declared in the API extensions, and the bounds must be non-negative integers that together define a non-empty range. The upper bound may also be **unlimited** in which case the number of instances depends upon the balance of create and delete requests issued for that type.

The auto-tester keeps track of the instances it creates for each resource type, in order to provide valid references and avoid too many 404 responses, and makes best efforts to keep the number of instances within any specified bounds.

For example, to place an upper bound on the number of **Customer** instances that should exist at any time:

```
Instances:
 Customer:
 maximum: 25
```

## InstanceDefault

The default instance bounds for any resource type not mentioned explicitly under **Instances**.

For example, to make the default unlimited:

```
InstanceDefault:
 minimum: 1
 maximum: unlimited
```

## Operations

---

Once the auto-tester has selected a resource type, it uses a similar weighting strategy to choose an operation from that type.

You control the frequency with which operations are selected by assigning them weights. Use the **Operations** key to assign weights to specific operations, and **OperationDefault** to set the default weight for any operations not specified.

A zero weight disables an operation entirely, and if all the operations are disabled for a resource then the resource itself is also disabled.

## Operations

The weights assigned to specific operations. The value is an object whose schema follows the pattern established by the **Paths** object in OpenAPI, but the values assigned to each operation are non-negative integer weights.

For example, to increase test focus on the **put\_customer** operation, and avoid **patch\_customer** completely:

```
Operations:
 /customers/{customer_id}:
 put: 10
 patch: 0
```

The special key **\_** can be used to match all methods defined on a path that are not called out explicitly. For example, to disable all but the **put\_customer** operation:

```
Operations:
 /customers/{customer_id}:
 put: 10
 _: 0
```

Remember that the weights are interpreted only across operations within the same resource type, and not across all operations in the API, so the overall distribution is determined by both the weight on the resource and the weight on the operation.

## OperationDefault

The default weight for any operations not mentioned explicitly under **Operations**. The value is an object where the keys are method names and the values are the default weights for those methods where they occur under any path. The special key **\_** matches all method names that are not listed separately.

For example, to disable all operations by default:

```
OperationDefault:
 _: 0
```

In this case there must be at least one non-zero weight under **Operations**.

## Datatypes

---

Place here any custom rules that guide the auto-tester in generating values for operation parameters and payloads. These rules override the default rules derived from the API definition and any semantic annotations in the API extensions.

Rule syntax is described in [Data Generation Rules](#).

## Schemas

Data generation rules for named schemas found under **definitions** (OpenAPI 2.0) or **components/schemas** (OpenAPI 3.0).

For example, given the schema definition:

```
Movie:
 type: object
 properties:
 title:
 type: string
 release_date:
 type: string
 format: date
 language:
 description: Two-letter ISO 639-1 language code
 type: string
 pattern: '[a-z]{2}'
```

The default generation rule for the **language** property, derived from this schema, will produce random string values that match the specified pattern (two lower-case alphabetic characters) but many of these will not be valid language codes, potentially giving rise to an excessive number of 400 responses.

To override the default generation rule and specify instead some particular language codes sufficient for testing:

```
Datatypes:
 schemas:
 Movie:
 properties:
 language:
 enum:
 - en
 - fr
 - de
```

To inject a small percentage of invalid values:

```
Datatypes:
 schemas:
 Movie:
 properties:
 language:
 choice:
 - enum:
 - en
 - fr
 - de
 weight: 98
 - const: xx
 weight: 2
```

## Operations

Data generation rules for operation parameters and payloads described by inline (or unnamed) schemas and types. The value is an object whose schema follows the pattern established by the **Paths** object in OpenAPI.

Note that the auto-tester only supports **path** and **query** parameters at present, and other parameter types are ignored.

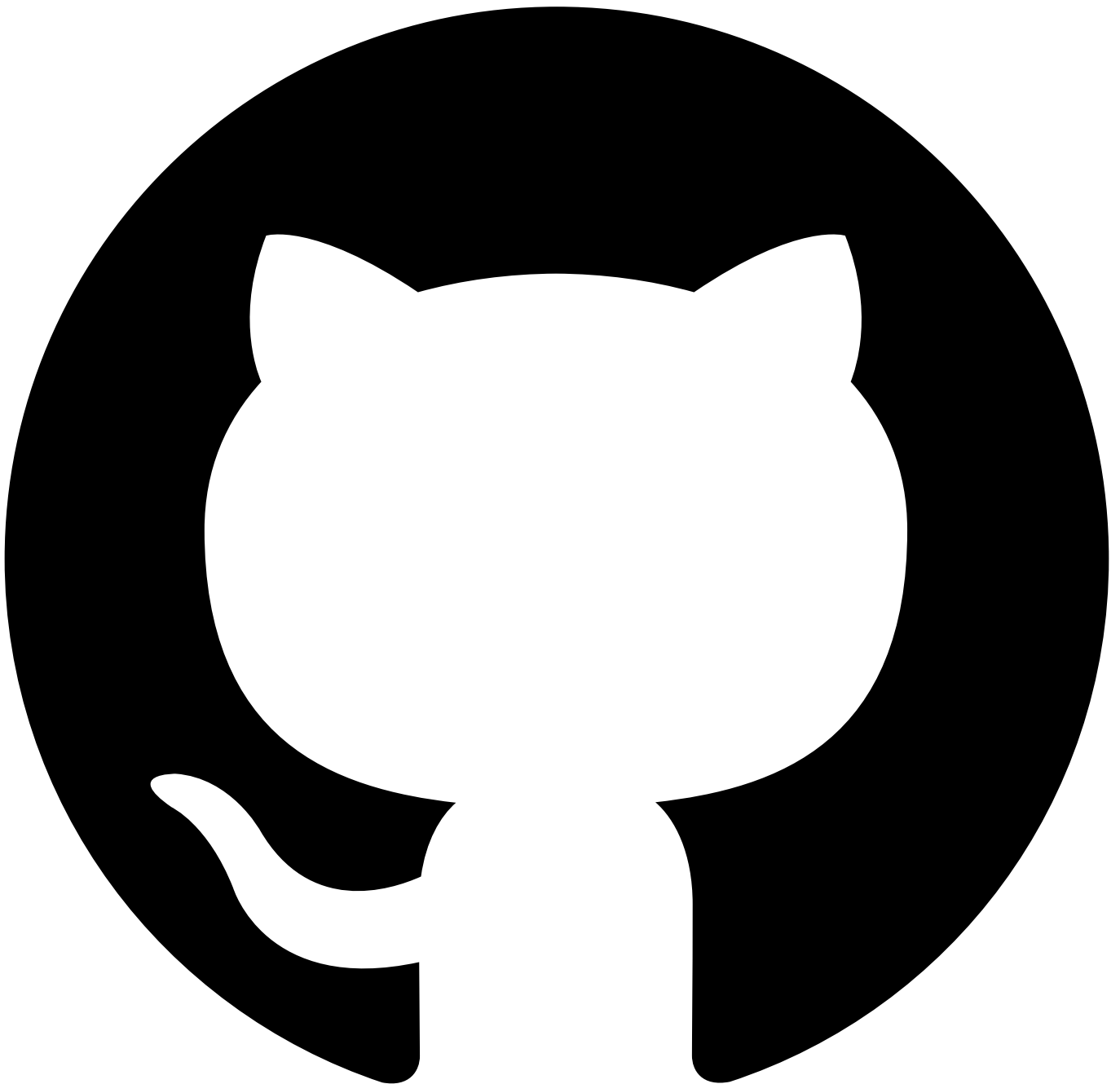
For example, given these (abbreviated) operation definitions:

```
paths:
 /movies:
 get:
 parameters:
 - name: limit
 in: query
 description: Sets an upper bound on the number of movies returned
 schema:
 type: integer
 post:
 requestBody:
 content:
 application/json:
 schema:
 type: object
 allOf:
 - $ref: '#/definitions/Movie'
 - properties:
 keywords:
 type: array
 items:
 type: string
```

To set values for the **limit** query parameter, and the **keywords** property of the payload to **post**:

```
Datatypes:
 operations:
 /movies:
 get:
 parameters:
 - name: limit
 in: query
 data:
 # pick a random value between 0 and 500 (inclusive)
 minimum: 0
 maximum: 500
```

```
post:
 requestBody:
 content:
 application/json:
 data:
 properties:
 keywords:
 # pick up to 3 items from the following list
 items:
 enum:
 - avant-garde
 - dystopia
 - epic
 - postmodern
 - remake
 - shootout
 minItems: 0
 maxItems: 3
```



Contribute in GitHub: [Edit online](#)

---

## Profile Security

Security is where you set credentials for the API under test. The auto-tester supports only **Basic** authentication (username and password) and **Bearer** (or **Token**) authentication; it does not support other authentication methods such as **OAuth**, or the use of custom headers.

Select your security type from the dropdown and enter the required information:

Security Type	Description	Credentials
None	Unsecured endpoint	n/a
Basic	Basic HTTP authentication	Username/Password
Basic-url	Same as <b>Basic</b> but omits the word <b>Basic</b> from the Authorization header	Username/Password
Bearer	Bearer (or Token) authentication	Token

If you provide incorrect credentials then they will not prevent the profile from running, but you can expect to see all requests fail, typically with the response **401 Unauthorized** or **403 Forbidden**.



If you provide a bearer token then be sure that it has a sufficient lifetime to cover the full period of the test, which is determined by the [stopping criteria](#) in the profile configuration. Otherwise, once the token expires, you can expect to see authentication failures as described earlier.

## Custom headers

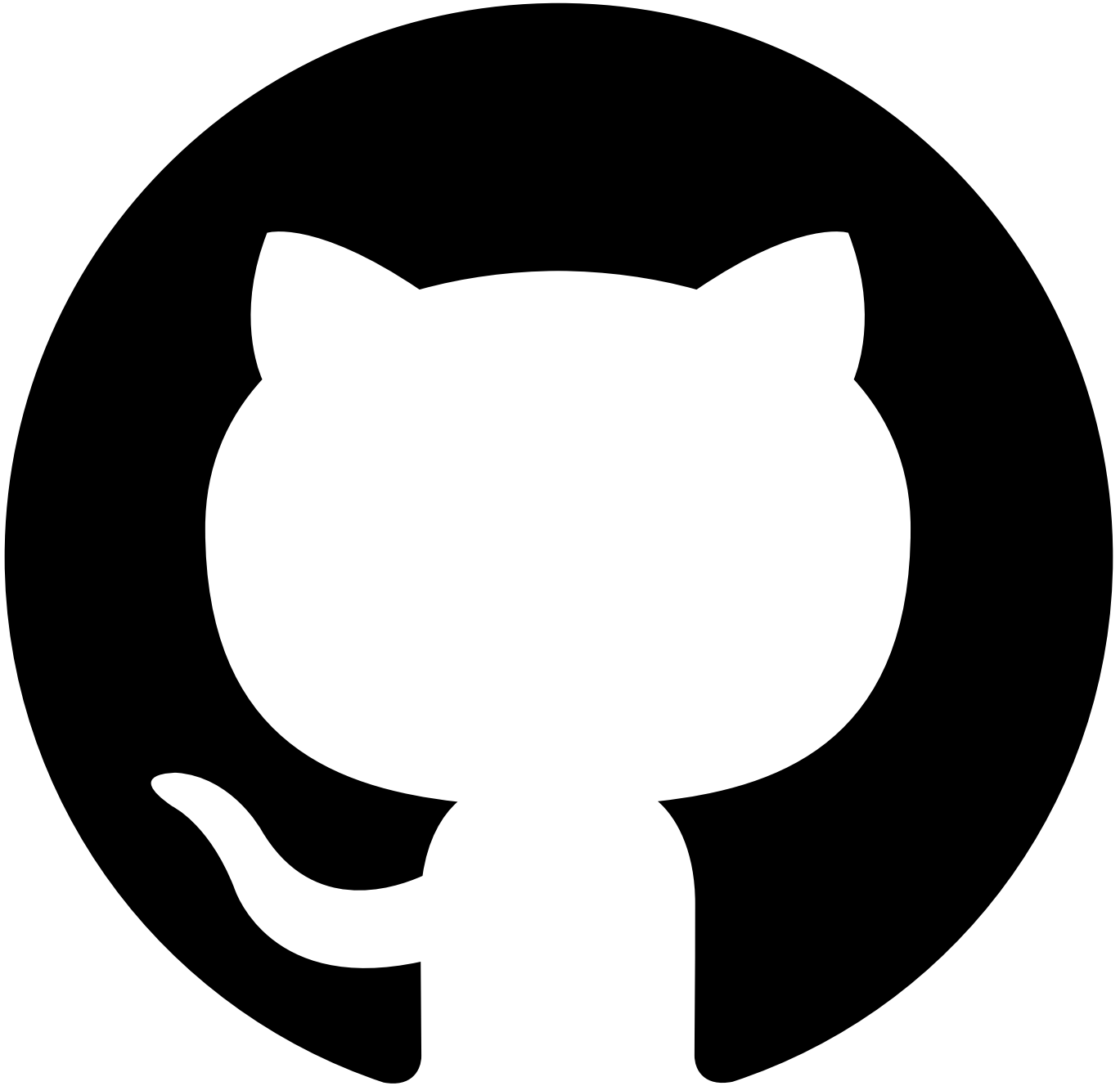
You can also configure custom security headers should your APIs require them for authentication/authorization. Simply add the required header(s) and the appropriate value(s) under the custom headers section like so:

### Custom headers

---

Name	Value	
X-IBM-Client-Id	.....	 

[Add new header](#)



Contribute in GitHub: [Edit online](#)

---

## API Extensions

An **API Extension** augments an OpenAPI definition with semantic information designed to improve the automatic generation of test cases. Extensions are used primarily by the AutoTest assistant to refine its randomly-generated test cases, but have wider applicability.

An extension comes in two parts:

- [Resources](#) identify the key resource types managed by the API, and their methods
- [Properties](#) attach semantic categories to schema properties, to constrain the format of values suggested for those fields

An extension is associated with a specific OpenAPI document and uses [JSON Pointer](#) notation to locate terms in that document.

---

## Resources

The **resources** section contains a set of resource definitions, where each definition contains at least the following:

- a name for the resource
- a pointer to a schema of type **object** that models the resource
- a collection of methods that operate on the resource

Typically, a resource definition also identifies a field within the schema that stores a unique identifier for instances of that type.

For example, consider this extract from an API extension:

```
resources:
 Book:
 schemas:
 primary:
 json_ptr: '#/definitions/Book'
 properties:
 id_name: '$.book_id'
 operations:
 create:
 - json_ptr: '#/paths/~1books/post'
 retrieve:
 - json_ptr: '#/paths/~1books~1{book_id}/get'
 - json_ptr: '#/paths/~1books/get'
 update:
 - json_ptr: '#/paths/~1books~1{book_id}/put'
 delete:
 - json_ptr: '#/paths/~1books~1{book_id}/delete'
```

The `resources` section contains a single resource definition with the following properties:

- it defines a resource named `Book`
- a book is modeled by a schema of the same name located in the OpenAPI document under `definitions`
- each book has a property `book_id` that holds its unique ID
- there are operations defined in the OpenAPI document under `paths` that create, retrieve, update and delete books

The correspondence between the resource name and the schema name is common but not required: a resource name can be anything, as long as it is unique within the API.

The unique ID field for a book is identified by a simplified form of JSON Path expression that locates a property (`book_id`) within the primary schema.

Operations (or methods) on a resource are grouped by category:

- Create
- Retrieve
- Update
- Delete

There can be multiple operations in each. In this example there are two operations that retrieve books: one that returns a single book by ID, and another that returns a list of books, and this is common. A different definition might have `patch` as well as (or instead of) `put` under `update`.

JSON Pointer expressions in a resource definition all refer implicitly to the associated OpenAPI document, so the prefix `#` introduces a relative location within that document. JSON Pointer syntax uses the string `~1` to stand for a forward slash character (`/`) to prevent confusion with the same character used as a separator, so the operation defined in the OpenAPI document by the term:

```
paths:
 /books/{book_id}:
 get:
```

is identified here by the JSON Pointer:

```
#/paths/~1books~1{book_id}/get
```

## Dependencies

A resource definition may also describe a dependency from one resource type to another.

For example:

```
Order:
 schemas:
 primary:
 json_ptr: '#/definitions/Order'
 properties:
 id_name: '$.order_id'
 dependencies:
 - name: Book
 required: true
 references:
 - name: $.book_id
 in: body
 dependee_deletion: disabled
 - name: Customer
 required: true
 references:
 - name: customer_id
 in: path
 dependee_deletion: mutual
 operations:
 create:
 - json_ptr: '#/paths/~1customers~1{customer_id}~1orders~1{order_id}/post'
 # rest omitted for brevity
```

This definition says that the resource `Order` has dependencies on two other resources `Customer` and `Book` (which must be defined in the same extension).

Both dependencies are **required** in the sense that an order cannot exist unless there is a customer (to place the order) and at least one book (to buy).

An order refers to an existing book through the property `book_id` in the primary schema, and to a customer through the path parameter `customer_id`, as in the `create` method shown (and in other methods omitted).

A customer *owns* a set of orders; this is a hierarchical relationship implicit in the path:

```
/customer/{customer_id}/orders/{order_id}
```

When a customer is deleted, all their orders are also deleted, and this behavior is indicated by the `mutual` value of the deletion property.

A different relationship applies to books: a book cannot be deleted until all the orders which refer to it have been deleted, so deletion is indicated as `disabled` in that case.

There are three possible values for the `dependee_deletion` property:

- `enabled`: a referenced resource can be deleted even while dependent resources that refer to it still exist
- `disabled`: a referenced resource cannot be deleted until all dependent resources that refer to it have been deleted
- `mutual`: deleting a referenced resource deletes all the dependent resources that refer to it: this is typical where the dependent is owned by or contained in the referent

## Pure resources

A pure resource is one that has no instances, or at least no instances that can be tracked with unique IDs, and is used as a placeholder for operations that do not behave as resource methods.

A pure resource differs from a standard resource in the following ways:

- it has no ID field (because there are no instances)
- dependencies have no `dependee_deletion` (for the same reason)
- all operations are grouped under a single `pure` category

For example:

```
ServiceAvailability:
 schemas:
 primary:
 json_ptr: '#/definitions/ServiceAvailability'
 operations:
 pure:
 - json_ptr: '#/paths/~1service~1status/get'
```

`ServiceAvailability` is a type that is returned as the result of a service status request but has no unique, persistent state.

## Properties

The `properties` section assigns semantic categories to schema properties occurring in operation payloads, to constrain the range of values interpolated into generated request bodies.

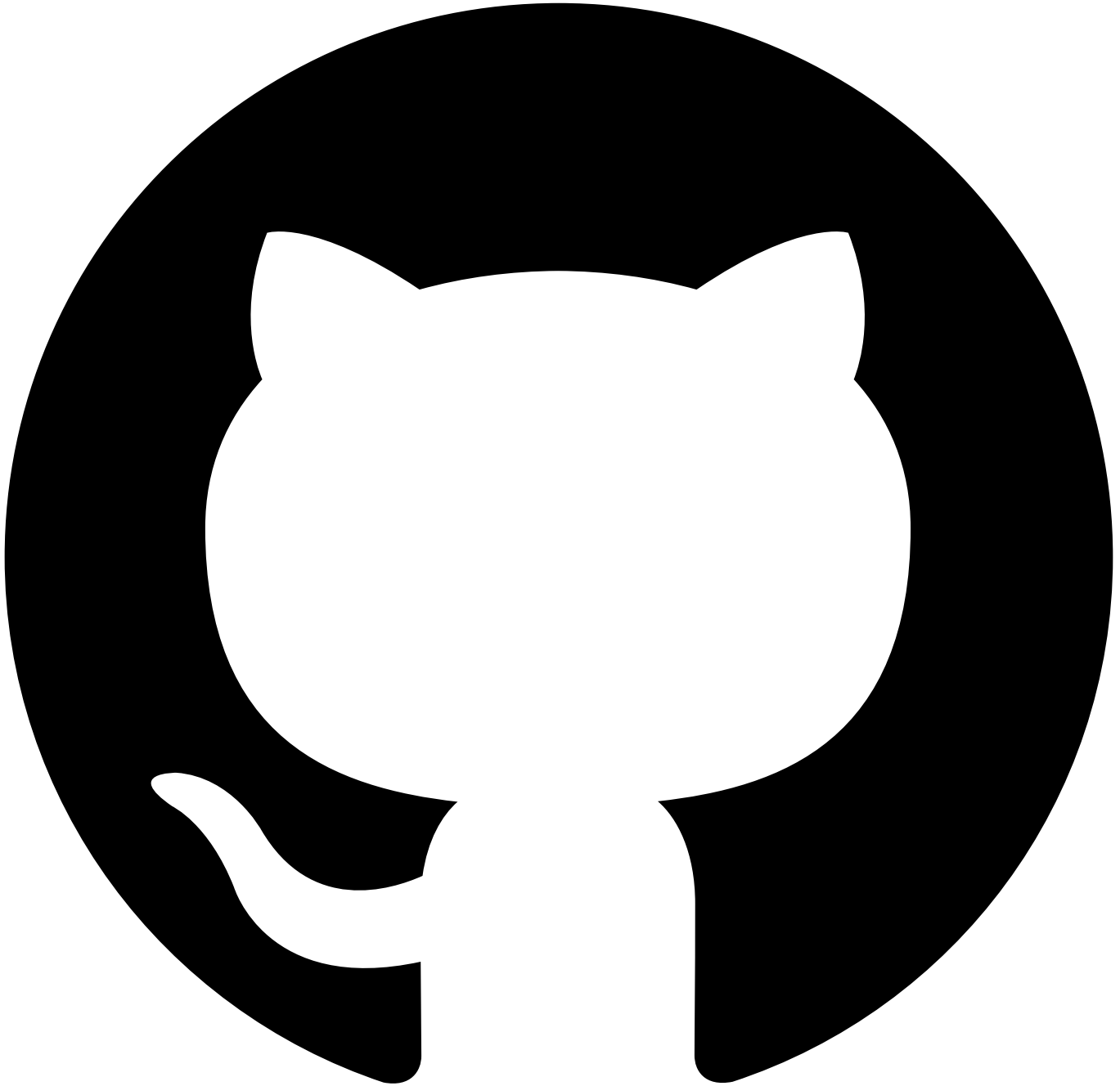
For example:

```
properties:
 - json_ptr: '#/definitions/CardPayment'
 items:
 - name: card_number
 semantic: credit_card_number
 - name: expiry_date
 semantic: expiry
 - name: security_code
 semantic: cvv
```

The JSON Pointer expression locates a schema named `CardPayment` in the API document under `definitions`; the schema must have `object` type with at least the properties `card_number`, `expiry_date` and `security_code` which are specified here to have particular semantics that will guide the auto-generation of appropriate values wherever a `CardPayment` occurs in a request payload.

Semantic categories must be drawn from the following set:

address	cvv	latitude	state_code
age	date	longitude	street
area_code	domain	minutes	temperature
birthday	email	month	time
certificate	expiry	name	timestamp
charset	first_name	number	token
cidr	gender	paragraph	twitter
city	geo_location	percentage	url
color	hours	phone	user_agent
content_encoding	humidity	phone_number	username
content_type	iban	prefix	uuid
coordinates	id	pressure	version
country	identity_provider	price	year
country_code	ip_address	revision	zip_code
credit_card_number	language	sentence	
currency	language_code	social_security_number	
currency_code	last_name	state	



Contribute in GitHub: [Edit online](#)

---

## Data Generation Rules

Data generation rules guide the auto-tester in choosing values to populate the parameters and payloads of its requests.

There are default rules derived from the API definition and extensions, but sometimes you want to override these, either to improve the general accuracy of the auto-tester or to tailor its requests to a specific use case, and you do that by adding custom rules to the [Datatypes](#) section of the profile configuration.

A data generation rule binds a **data generator** to a schema or type in the API definition such that, when the auto-tester assembles the data for a request and encounters that schema or type, it uses the generator to provide an appropriate value at that point.

---

## Data generators

A **data generator** describes a never-ending sequence of values, and whenever the auto-tester calls on a generator it takes the next value from the sequence.

The syntax for generation rules reuses terms from JSON Schema where the implied constraints on a schema are sufficient to guide the auto-tester in generating conformant values. Such constraints occurring in the API definition help inform the default rules.

### Constant



The **constant** generator always returns the same value, which can be of any simple type.

```
const: 0
==> 0, 0, 0, ...

const: Thursday
==> "Thursday", "Thursday", "Thursday", ...
```

## Enumeration

The **enumeration** generator returns values selected at random from a given list, which can be of any simple type.

```
enum:
- 0
- 1
- 2
==> 1, 0, 2, 0, ...

enum:
- Thursday
- Friday
- Saturday
==> "Friday", "Saturday", "Saturday", ...
```

## Pattern

The **pattern** generator returns string values that match a given regular expression.

```
pattern: '[0-9]{3}'
==> "884", "924", "121", ...

pattern: '[a-z][a-z_]{0,15}'
==> "dbe", "kg_cfsv", "dhhfbgbjxavmsaxq", ...
```

## Range

The **range** generator returns numeric values uniformly distributed over a given range (inclusive).

```
minimum: 0
maximum: 10
==> 10, 6, 8, ...

minimum: 0
==> 10029625, 3162517758, 841054690, ...
```

## Semantic

The **semantic** generator returns values that conform to some pre-defined semantic category, covering names, addresses, credit card numbers, etc. The set of categories is the same as that allowed in the [API extensions](#).

```
semantic: email
==> "herbertschaden@fahey.net", "emmyfarrell@cruickshank.com", "vilmaprohaska@zboncak.org", ...

semantic: first_name
==> "Antwan", "Connie", "Modesto", ...
```

## Resource

The **resource** generator returns values that are unique identifiers of resource instances of a given type, selected at random from the pool of existing instances. The identifier of an instance is taken from the `id_name` property of the resource type defined in the [API extensions](#) and its type and format will vary accordingly.

```
resource: Customer
==> "e5537298-9041-47fc-b97e-c910d8f9d971", "f60521c8-2ccb-4036-a742-9f350271675f", "e5537298-9041-47fc-b97e-c910d8f9d971", ...
```

## Array

The **array** generator returns array values whose contents are populated from a data generator defined recursively. The length of the array is chosen at random between `minItems` and `maxItems` inclusive.

```
items:
 pattern: '[0-9]{3}'
minItems: 3
maxItems: 3
==> ["770", "496", "219"], ["431", "895", "331"], ["864", "130", "204"], ...

items:
 enum:
 - 0
 - 1
minItems: 0
```

```
maxItems: 5
```

```
==> [], [1, 0, 1, 1], [0], ...
```

## Object

The **object** generator returns object values whose properties are populated from named data generators, defined recursively.

```
properties:
```

```
 x:
 minimum: 0
 maximum: 100
 y:
 minimum: 0
 maximum: 100
```

```
==> {"x": 59, "y": 63}, {"x": 22, "y": 8}, {"x": 95, "y": 57}, ...
```

```
properties:
```

```
 code:
 pattern: '[0-9]{3}'
 language:
 enum:
 - en
 - fr
 - de
```

```
==> {"code": "452", "language": "fr"}, {"code": "504", "language": "en"}, {"code": "846", "language": "fr"}, ...
```

If a property is not required by the target schema then it may be marked with an **optional** attribute which specifies a frequency (between 0.0 and 1.0) that determines how often the property is included in the generated object (where 0 means never, and 1 means always).

```
properties:
```

```
 code:
 pattern: '[0-9]{3}'
 language:
 optional: 0.5
 enum:
 - en
 - fr
 - de
 tag:
 optional: 0.0
```

```
==> {"code": "452", "language": "fr"}, {"code": "504", "language": "en"}, {"code": "846"}, ...
```

## Choice

The **choice** generator returns values selected at random from a given list of data generators, using optional weights to bias the outcome. The alternatives must be simple generators, which excludes **array** and **object** generators.

```
choice:
```

```
 # bias the selection towards English
 - const: en
 weight: 5
 - const: fr
 - const: de
```

```
==> "en", "en", "en", ..., "de", "en", ...
```

```
choice:
```

```
 # normal values in the range 1-10
 - minimum: 1
 maximum: 10
 weight: 98
 # with a small percentage of outliers
 - const: 999
 weight: 2
```

```
==> 6, 3, 9, 1, ..., 999, 6, ...
```

## Data generation rules

A **data generation rule** binds a data generator to a schema occurring in the API definition so that the auto-tester knows how to generate values for that schema in an API call. Rules are specified in the [Datatypes](#) section of the profile configuration, and override the default rules derived from the API definition and extensions.

Schemas occur widely in the OpenAPI specification, but there are three key locations which have most relevance to the auto-tester where the profile allows override rules:

- Named schemas
- Parameters
- Inline request bodies

### Named schemas

A schema rule assigns a data generator to a named schema from the API definition, found either under **definitions** (OpenAPI 2.0) or **components/schema** (OpenAPI 3.0). The auto-tester will use the specified generator wherever the named schema occurs in a request.

Schema rules appear in the [schemas](#) section of the profile configuration, under **Datatypes**.

Because this is an *override*, the generator may be incomplete for the original schema, and the auto-tester will fall back to using the default rules to cover any gaps. In particular, if the schema describes an object type then the generator need only name a subset of the object properties, and the auto-tester will apply the override for those properties but use default rules for the rest.

For example, given this named schema in the API definition:

```
Movie:
 type: object
 properties:
 title:
 type: string
 release_date:
 type: string
 format: date
 language:
 description: Two-letter ISO 639-1 language code
 type: string
 pattern: '[a-z]{2}'
```

The default rule derived from the schema returns values such as:

```
{
 "title": "mkLHM22Ohabb6YG4-wdZFUwxQa7dn",
 "release_date": "2018-09-14",
 "language": "jq"
}
```

You can refine this by overriding the default rules for the `title` and `language` properties:

```
Datatypes:
 schemas:
 Movie:
 properties:
 title:
 semantic: sentence
 language:
 enum:
 - en
 - fr
 - de
```

This rule binds an object generator (introduced by the keyword `properties`) to the schema named `Movie` which -- combined with the default rule for `release_date` -- returns values such as:

```
{
 "title": "What dog everybody am myself hourly meeting how group over example her",
 "release_date": "2094-06-31",
 "language": "de"
}
```

## Parameters

A parameter rule binds a data generator to a named parameter within an operation found under `paths` in the API definition.

Parameter rules appear in the [operations](#) section of the profile configuration, under `Datatypes`.

For example, given this parameter definition on `GET /movies` in the API definition that restricts the number of movies returned in a single call:

```
paths:
 /movies:
 get:
 parameters:
 - name: limit
 in: query
 description: Sets an upper bound on the number of movies returned
 schema:
 type: integer
```

The default rule derived from this definition will return values from the full integer range, but you can override this with a rule in the profile configuration that specifies a more realistic range, with occasional outliers to check the validation logic:

```
Datatypes:
 operations:
 /movies:
 get:
 parameters:
 - name: limit
 data:
 choice:
 - minimum: 1
 maximum: 500
 weight: 95
 - const: 0
 weight: 3
 - const: 8000000
 weight: 2
```

The format of the rule follows the format of the original definition but with the keyword `data` in place of `schema` to introduce the data generator.

In OpenAPI, a parameter is uniquely identified by a name *and* a location (so the same name could be used in, say, both query and header) so the parameter rules allow the `in` property as optional where needed to disambiguate. It's not used in this example because the query parameter is the only one named `limit`.

The auto-tester only supports parameters of type `query` and `path` at present (and `body` in OpenAPI 2.0). Rules bound to other parameter types are ignored.

## Request body

A request body rule binds a data generator to the body of a request where the corresponding schema is specified directly as part of the operation in the API definition, rather than via a reference to a named schema.

Request body rules appear in the [operations](#) section of the profile configuration, under **Datatypes**.

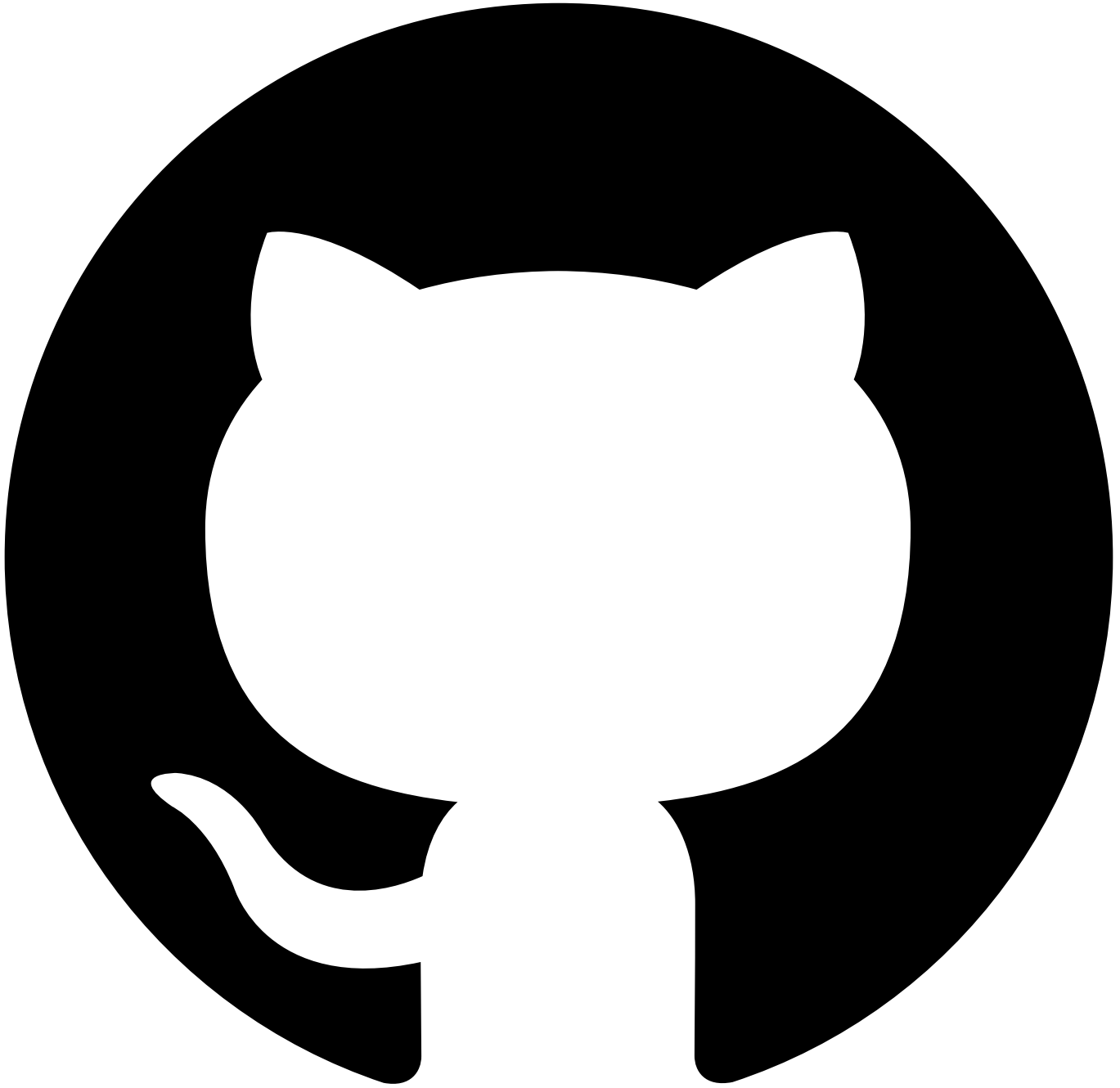
For example, given this definition of the payload to a **POST** request:

```
paths:
 /movies:
 post:
 requestBody:
 content:
 application/json:
 schema:
 type: object
 allOf:
 - $ref: '#/components/schemas/Movie'
 - properties:
 keywords:
 type: array
 items:
 type: string
```

The following rule generates some appropriate keywords to attach to a new movie (where the movie itself is covered by the rules discussed earlier):

```
Datatypes:
 operations:
 /movies:
 post:
 requestBody:
 content:
 application/json:
 data:
 properties:
 keywords:
 # pick up to 3 items from the following list
 items:
 enum:
 - avant-garde
 - dystopia
 - epic
 - postmodern
 - remake
 - shootout
 minItems: 0
 maxItems: 3
```

As with the parameter rules, the format follows that of the original definition but with **data** in place of **schema**. The definition is in the style of OpenAPI 3.0, but the rule can work for OpenAPI 2.0 as well as long as the content type is consistent; alternatively, starting from an OpenAPI 2.0 definition, you can use a **body** parameter rule to specify the payload, but that does not work for 3.0.



Contribute in GitHub: [Edit online](#)

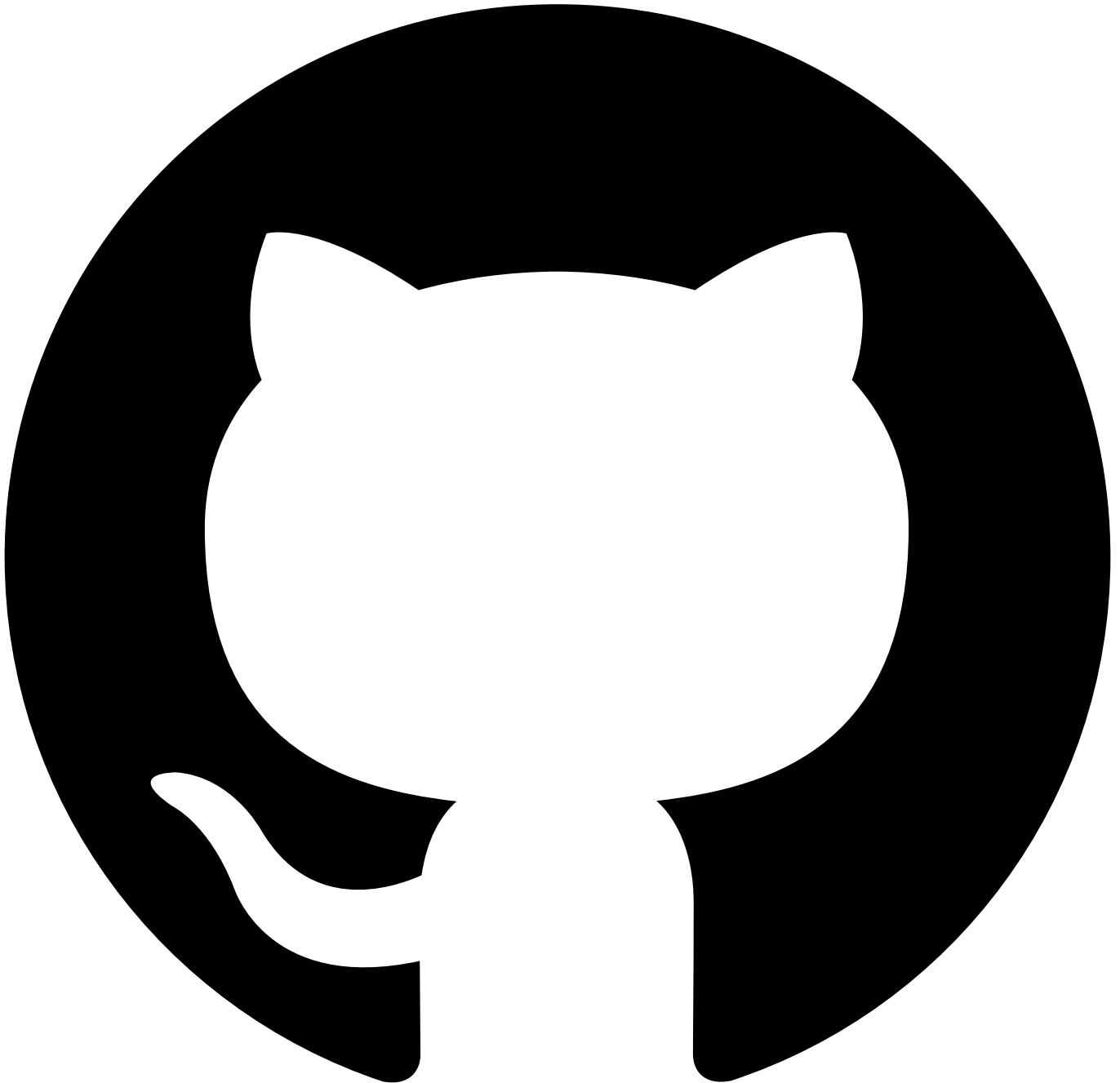
---

## Assertion components

IBM API Connect Test APIs supports the following assertion types:

- [Assert Compares](#)  
Asserts that two elements are equivalent in some way.
- [Assert Contains](#)  
Asserts that the value of the element identified by a given expression contains a specific substring.
- [Assert Equals](#)  
Asserts that the value of the element identified by a given expression is equal to a specific value.
- [Assert Exists](#)  
Asserts that the value of the element identified by a given expression exists.
- [Assert Greater](#)  
Asserts that the value of the element identified by a given expression is greater than a specific value.

- [Assert In](#)  
Asserts that the element identified by a given expression matches at least one item from a given list.
- [Assert Is](#)  
Asserts that the value of the element identified by a given expression is of a specific type.
- [Assert Less](#)  
Asserts that the value of the element identified by a given expression is less than a specific value.
- [Assert Matches](#)  
Asserts that the value of the element identified by a given expression matches a knowledge base of some kind.
- [Set](#) Set a variable during test executions



Contribute in GitHub: [Edit online](#)

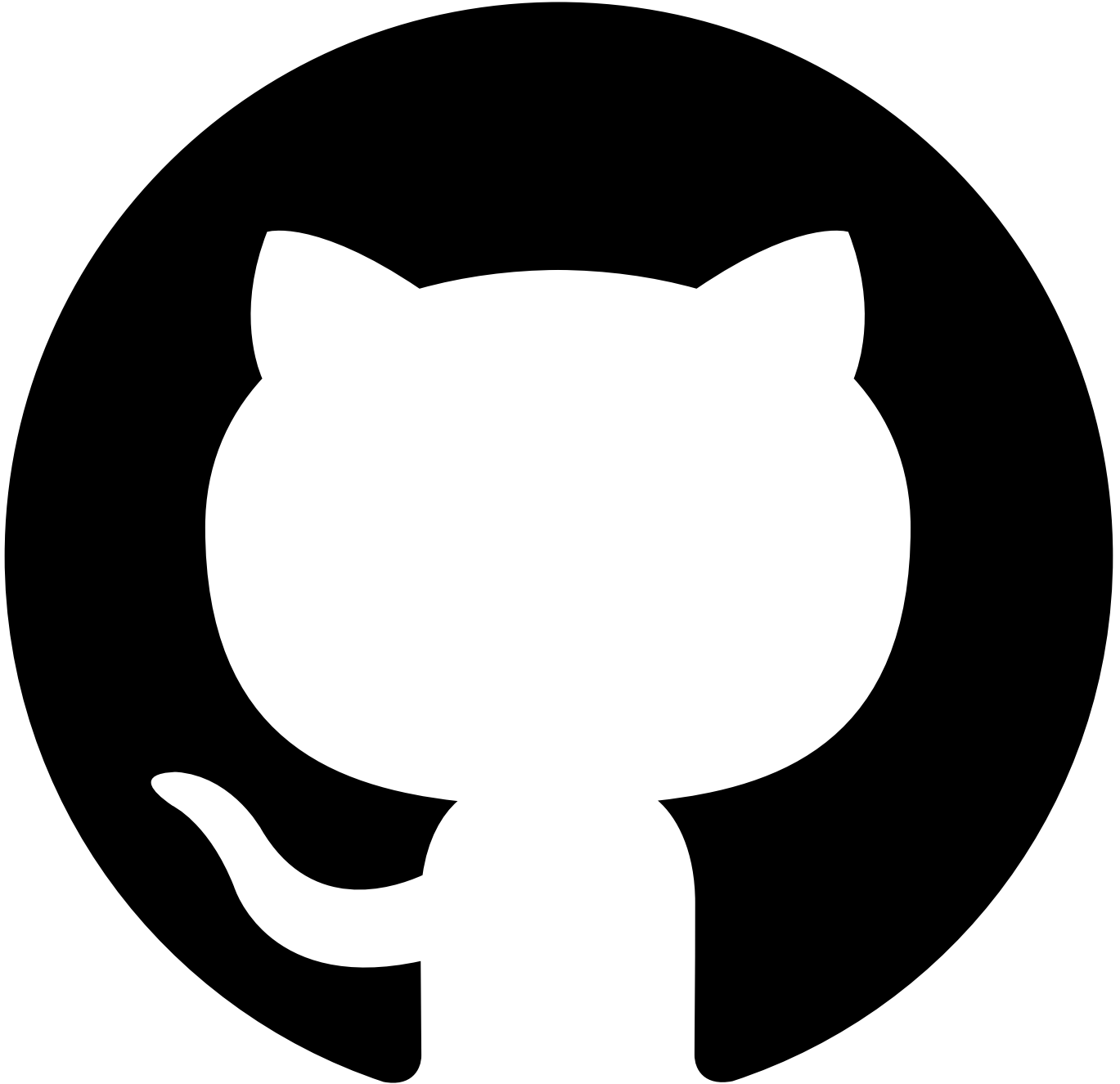
---

## Assert Compares

Asserts that two elements are equivalent in some way.

## Parameters

Name	Type	Required?	Description	Default value
Expression 1	Expression	Yes	Path to the first element for comparison; for example, <code>payload1.productId</code> .	
Expression 2	Expression	Yes	Path to the second element for comparison; for example, <code>payload2.productId</code> .	
Mode	Selection list	Yes	Select one of the following values: <ul style="list-style-type: none"><li>• <b>text</b>: compare the text of the two elements as plain text.</li><li>• <b>values</b>: compare the two elements regardless of their text layout.</li><li>• <b>structure</b>: compare only the structure of the two elements.</li></ul>	text
Level	Selection list	No	Select <b>error</b> or <b>warning</b> . Specifies whether, if the assertion fails, it should be considered an 'error' or just a 'warning.' A warning does not trigger alerts, such as email or text messages.	error
Stop test if fails	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the test is immediately stopped if the assertion fails.	false
Assertion comment	String	No	An optional comment for information.	



Contribute in GitHub: [Edit online](#)

---

## Assert Contains

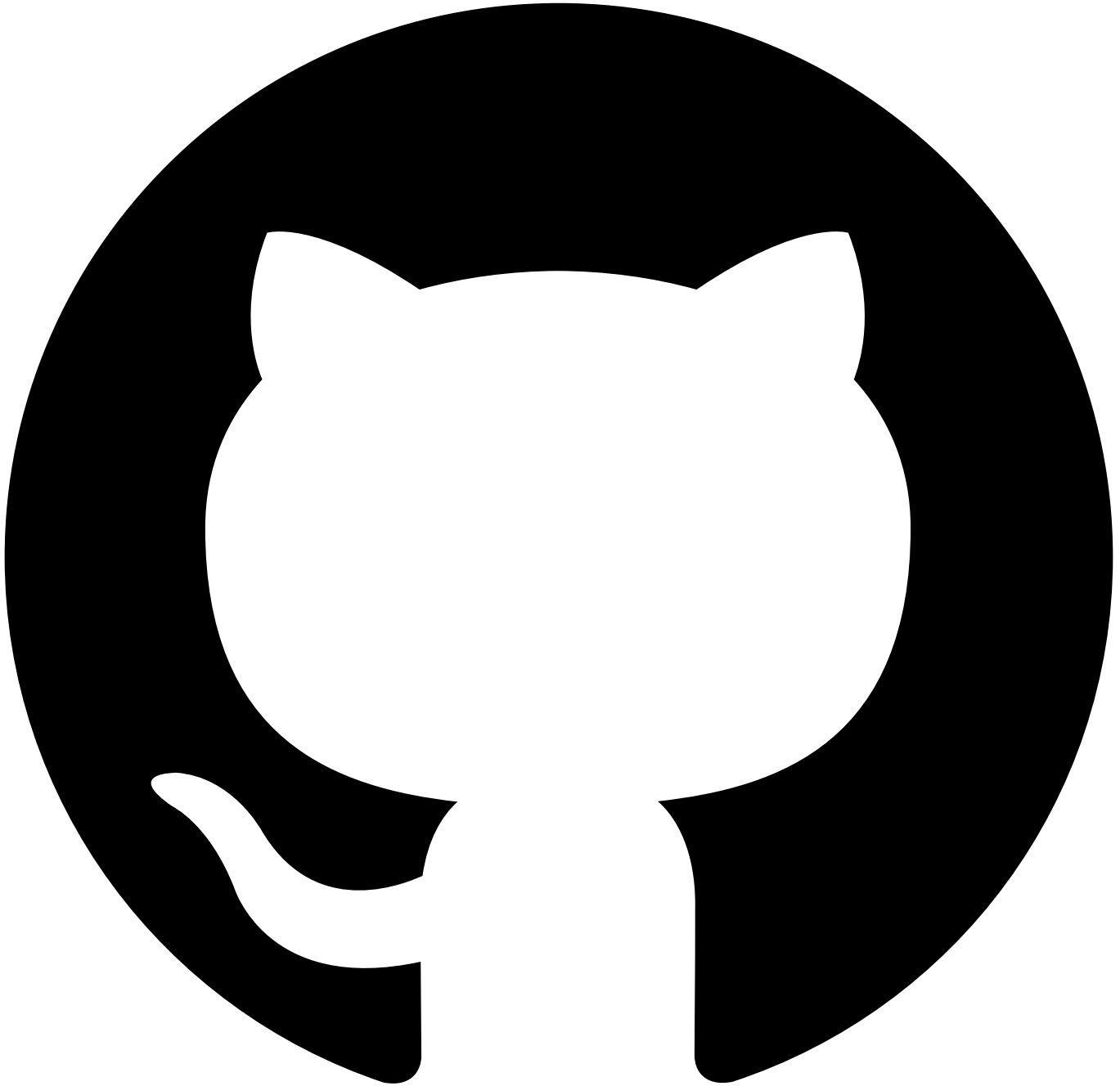
Asserts that the value of the element identified by a given expression contains a specific substring.

---

### Parameters

Name	Type	Required?	Description	Default value
Expression	Expression	Yes	Path to the element to be searched; for example, <code>payload.productId</code> .	
Value	String	Yes	Path to the second element for comparison; for example, <code>payload2.productId</code> .	
Mode	Selection list	No	Select one of the following values: <ul style="list-style-type: none"><li><b>all</b>: If the path expression matches multiple elements in the payload then they must all match the assertion.</li><li><b>one</b>: Only one of the matching elements in the payload needs to match the assertion.</li></ul>	all
Level	Selection list	No	Select <b>error</b> or <b>warning</b> . Specifies whether, if the assertion fails, it should be considered an 'error' or just a 'warning.' A warning does not trigger alerts, such as email or text messages.	error
Modifier	Selection list	No	Select <b>not</b> to negate the assertion; the assertion is considered verified if it does not pass.	None
Execute if item exists	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the assertion is evaluated only if the element exists. This is useful when the element does not always exist.	false
Stop test if fails	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the test is immediately stopped if the assertion fails.	false
Assertion comment	String	No	An optional comment for information.	





Contribute in GitHub: [Edit online](#)

# Assert Equals

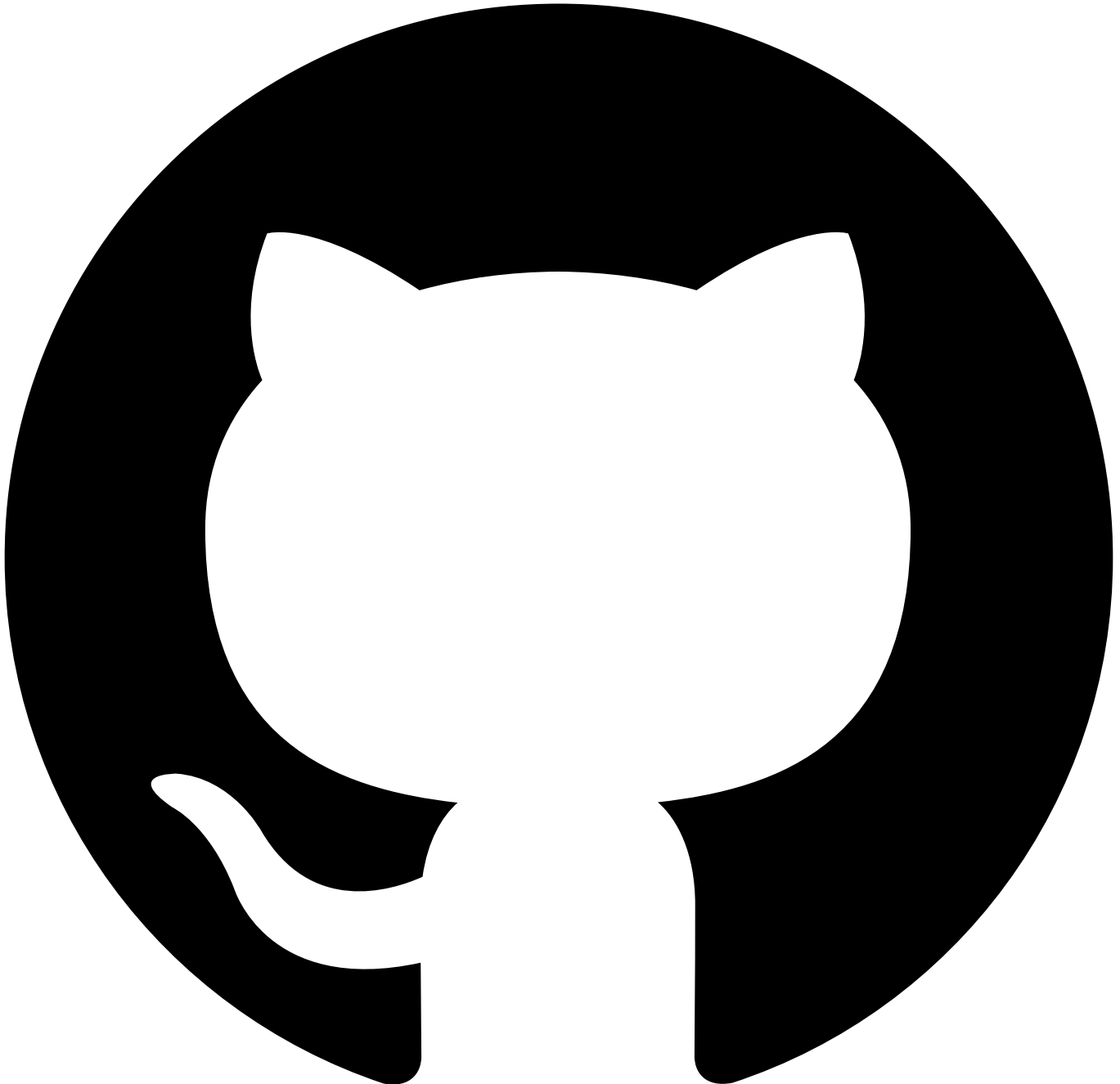
Asserts that the value of the element identified by a given expression is equal to a specific value.

---

## Parameters

Name	Type	Required?	Description	Default value
Expression	Expression	Yes	Path to the element to be compared with the specified value; for example, <code>payload.productId</code> .	
Value	String	Yes	The value that the element is to be compared with.	
Type	Selection list	No	The data type of the value. If you do not specify a data type, the values are compared as strings. Select one of the following values: <ul style="list-style-type: none"><li><b>integer</b>: the values are evaluated as whole numbers.</li><li><b>float</b>: the values are evaluated as decimal numbers.</li></ul>	

Name	Type	Required?	Description	Default value
Mode	Selection list	No	Select one of the following values: <ul style="list-style-type: none"> <li><b>all:</b> If the path expression matches multiple elements in the payload then they must all match the assertion.</li> <li><b>one:</b> Only one of the matching elements in the payload needs to match the assertion.</li> </ul>	all
Level	Selection list	No	Select <b>error</b> or <b>warning</b> . Specifies whether, if the assertion fails, it should be considered an 'error' or just a 'warning.' A warning does not trigger alerts, such as email or text messages.	error
Modifier	Selection list	No	Select <b>not</b> to negate the assertion; the assertion is considered verified if it does not pass.	None
Execute if item exists	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the assertion is evaluated only if the element exists. This is useful when the element does not always exist.	false
Stop test if fails	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the test is immediately stopped if the assertion fails.	false
Assertion comment	String	No	An optional comment for information.	



Contribute in GitHub: [Edit online](#)

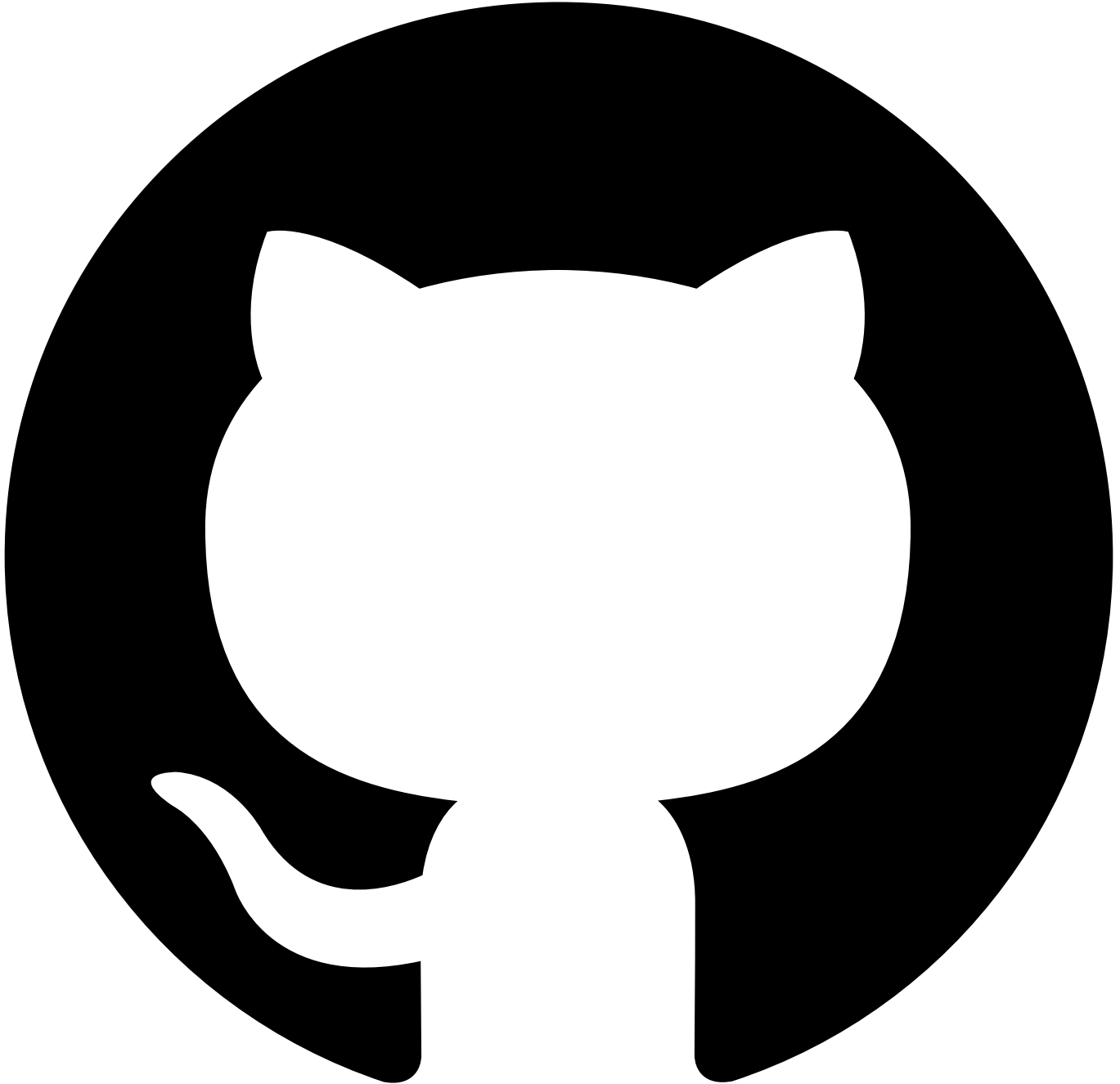
---

## Assert Exists

Asserts that the value of the element identified by a given expression exists. The presence of the element, even if it is empty, is enough to consider it a valid assertion.

## Parameters

Name	Type	Required?	Description	Default value
Expression	Expression	Yes	Path to the element to be checked for existence; for example, <code>payload.productId</code> .	
Mode	Selection list	No	Select one of the following values: <ul style="list-style-type: none"><li>• <b>all</b>: If the path expression matches multiple elements in the payload then they must all match the assertion.</li><li>• <b>one</b>: Only one of the matching elements in the payload needs to match the assertion.</li></ul>	all
Level	Selection list	No	Select <b>error</b> or <b>warning</b> . Specifies whether, if the assertion fails, it should be considered an 'error' or just a 'warning.' A warning does not trigger alerts, such as email or text messages.	error
Modifier	Selection list	No	Select <b>not</b> to negate the assertion; the assertion is considered verified if it does not pass.	None
Stop test if fails	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the test is immediately stopped if the assertion fails.	false
Assertion comment	String	No	An optional comment for information.	



Contribute in GitHub: [Edit online](#)

---

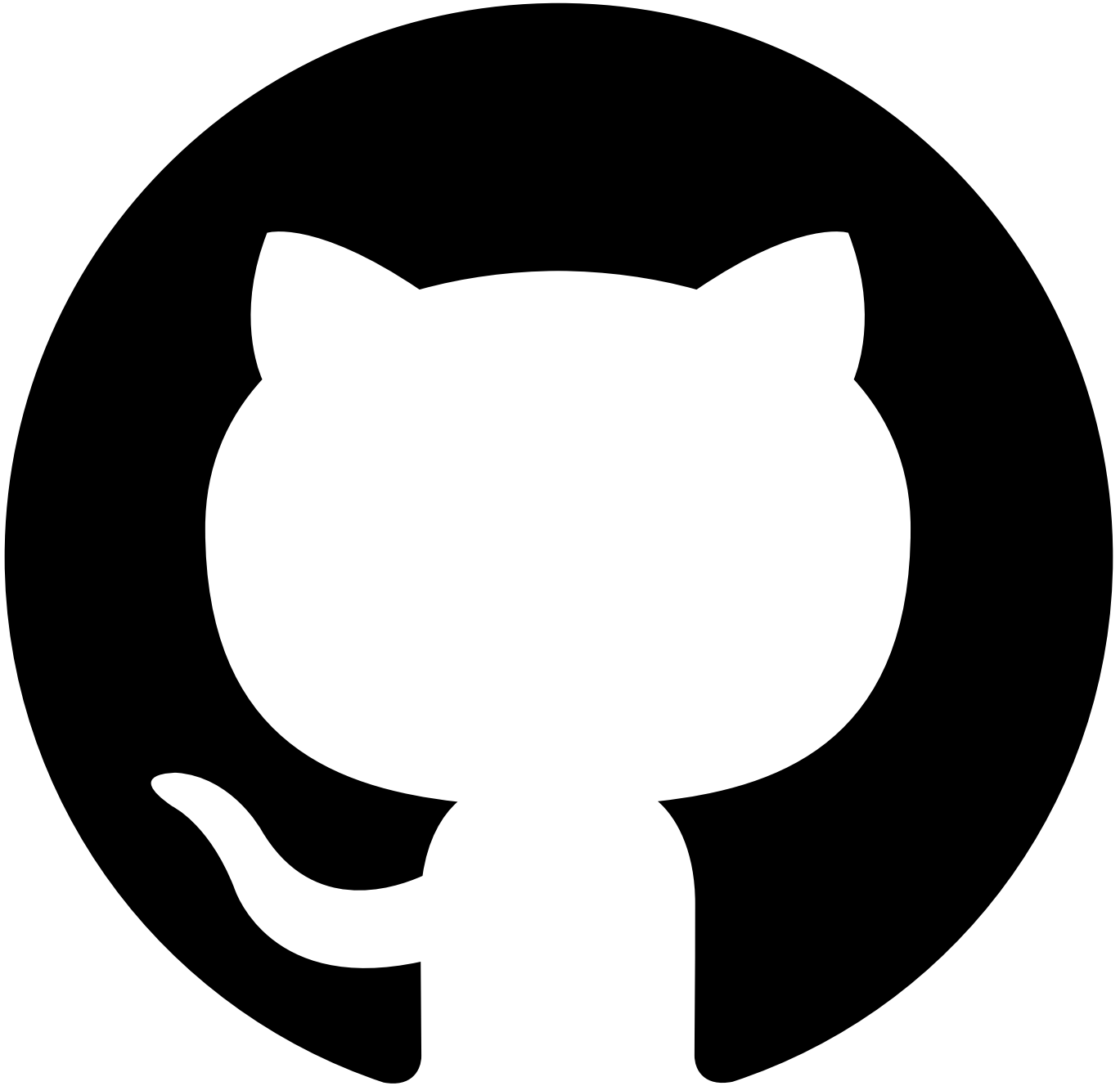
## Assert Greater

Asserts that the value of the element identified by a given expression is greater than a specific value.

---

### Parameters

Name	Type	Required?	Description	Default value
Expression	Expression	Yes	Path to the element to be compared with the specified value; for example, <code>payload.productId</code> .	
Value	String	Yes	The value that the element is to be compared with.	
Type	Selection list	No	The data type of the value. If you do not specify a data type, the values are compared as strings. Select one of the following values: <ul style="list-style-type: none"><li>• <b>integer</b>: the values are evaluated as whole numbers.</li><li>• <b>float</b>: the values are evaluated as decimal numbers.</li></ul>	
Mode	Selection list	No	Select one of the following values: <ul style="list-style-type: none"><li>• <b>all</b>: If the path expression matches multiple elements in the payload then they must all match the assertion.</li><li>• <b>one</b>: Only one of the matching elements in the payload needs to match the assertion.</li></ul>	all
Level	Selection list	No	Select <b>error</b> or <b>warning</b> . Specifies whether, if the assertion fails, it should be considered an 'error' or just a 'warning.' A warning does not trigger alerts, such as email or text messages.	error
Modifier	Selection list	No	Select <b>not</b> to negate the assertion; the assertion is considered verified if it does not pass.	None
Execute if item exists	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the assertion is evaluated only if the element exists. This is useful when the element does not always exist.	false
Stop test if fails	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the test is immediately stopped if the assertion fails.	false
Assertion comment	String	No	An optional comment for information.	



Contribute in GitHub: [Edit online](#)

# Assert In

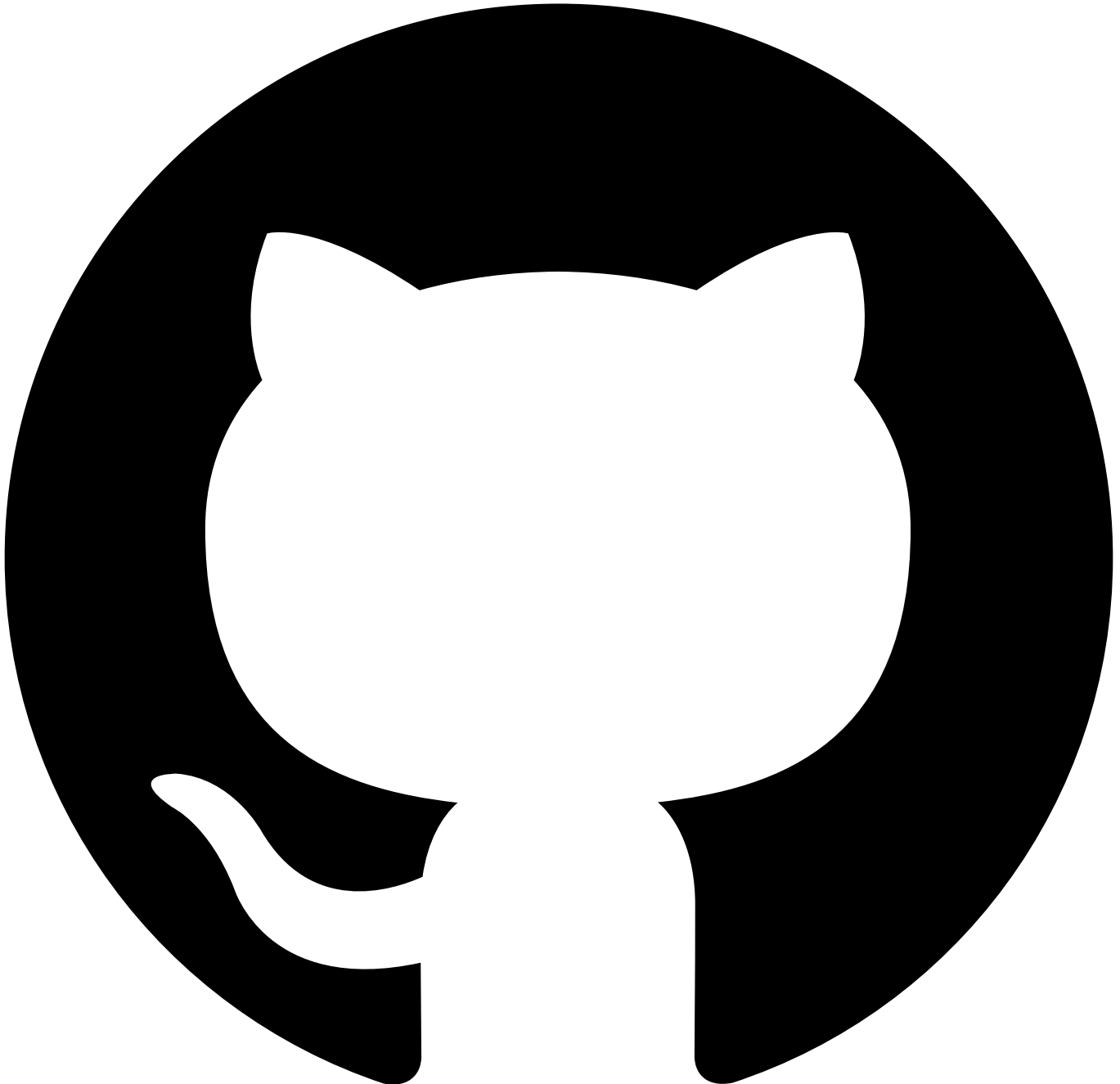
Asserts that the element identified by a given expression matches at least one item from a given list.

---

## Parameters

Name	Type	Required?	Description	Default value
Expression	Expression	Yes	Path to the element to be compared with the items in the list; for example <code>payload.productId</code> .	
Value	String	Yes	The items that the specified element is to be compared with. Click anywhere in the field, enter the first item, then click the + icon to add further items. Alternatively, you can enter the items as a comma separated list.	
Type	Selection list	No	The data type of the value. If you do not specify a data type, the values are compared as strings. Select one of the following values: <ul style="list-style-type: none"><li><b>integer</b>: the values are evaluated as whole numbers.</li><li><b>float</b>: the values are evaluated as decimal numbers.</li></ul>	

Name	Type	Required?	Description	Default value
Mode	Selection list	No	Select one of the following values: <ul style="list-style-type: none"> <li><b>all</b>: If the path expression matches multiple elements in the payload then they must all match the assertion.</li> <li><b>one</b>: Only one of the matching elements in the payload needs to match the assertion.</li> </ul>	all
Level	Selection list	No	Select <b>error</b> or <b>warning</b> . Specifies whether, if the assertion fails, it should be considered an 'error' or just a 'warning.' A warning does not trigger alerts, such as email or text messages.	error
Modifier	Selection list	No	Select <b>not</b> to negate the assertion; the assertion is considered verified if it does not pass.	None
Execute if item exists	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the assertion is evaluated only if the element exists. This is useful when the element does not always exist.	false
Stop test if fails	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the test is immediately stopped if the assertion fails.	false
Assertion comment	String	No	An optional comment for information.	



Contribute in GitHub: [Edit online](#)

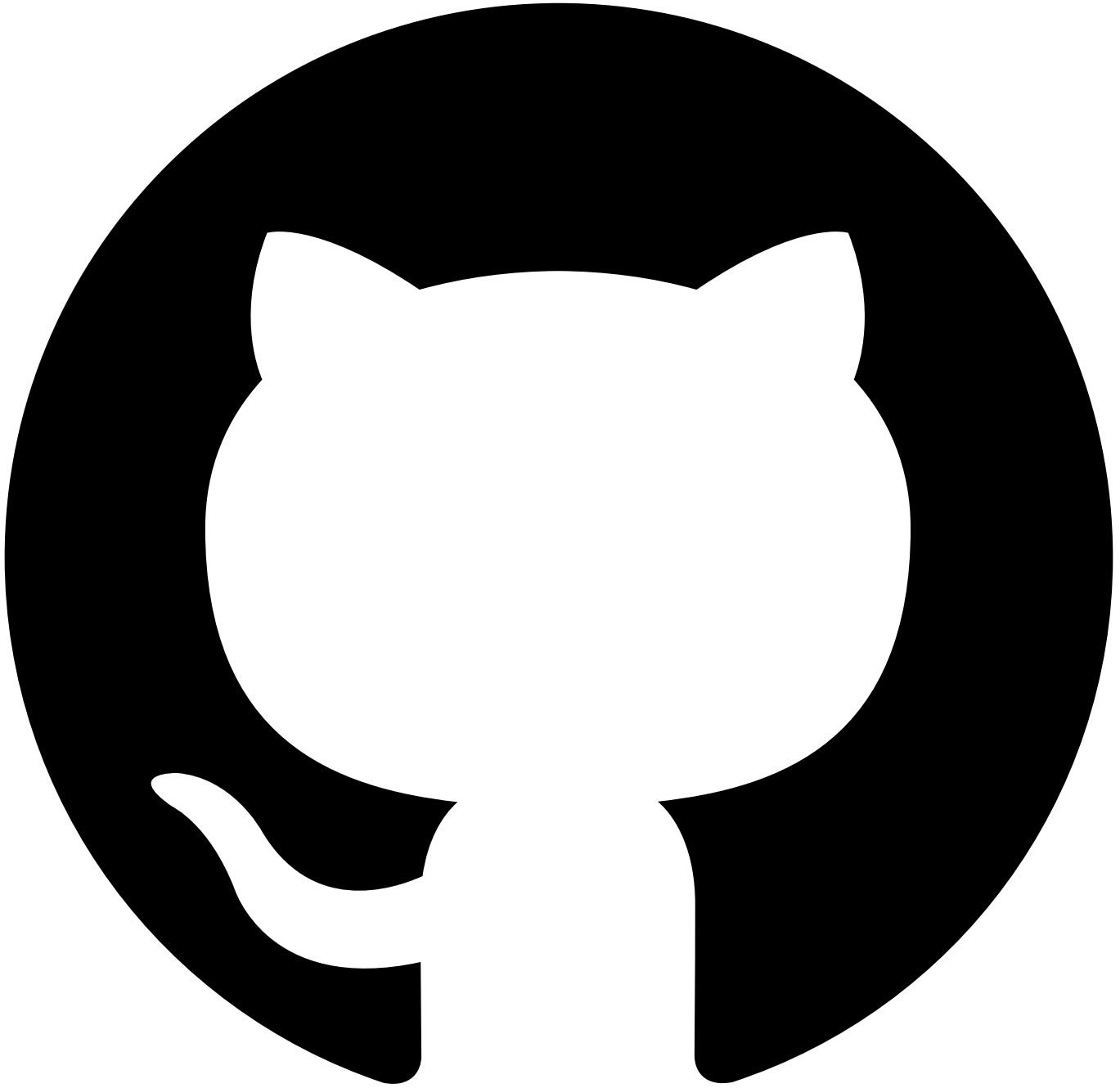
---

## Assert Is

Asserts that the value of the element identified by a given expression is of a specific type.

## Parameters

Name	Type	Required?	Description	Default value
Expression	Expression	Yes	Path to the element whose type is to be examined; for example, <code>payload.productId</code> .	
Type	Selection list	Yes	Select one of the following values: <ul style="list-style-type: none"> <li>• <b>integer</b>: Checks if the element is an integer value.</li> <li>• <b>float</b>: Checks if the element is a decimal value.</li> <li>• <b>url</b>: Checks if the element is a well-formatted URL.</li> <li>• <b>boolean</b>: Checks if the element is a boolean value.</li> <li>• <b>phone</b>: Checks if the element is a valid phone number format.</li> <li>• <b>email</b>: checks if the element is a valid email format.</li> <li>• <b>map</b>: Checks if the element is a map type.</li> <li>• <b>array</b>: Checks if the element is an array.</li> </ul>	
Mode	Selection list	No	Select one of the following values: <ul style="list-style-type: none"> <li>• <b>all</b>: If the path expression matches multiple elements in the payload then they must all match the assertion.</li> <li>• <b>one</b>: Only one of the matching elements in the payload needs to match the assertion.</li> </ul>	all
Level	Selection list	No	Select <b>error</b> or <b>warning</b> . Specifies whether, if the assertion fails, it should be considered an 'error' or just a 'warning.' A warning does not trigger alerts, such as email or text messages.	error
Modifier	Selection list	No	Select <b>not</b> to negate the assertion; the assertion is considered verified if it does not pass.	None
Execute if item exists	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the assertion is evaluated only if the element exists. This is useful when the element does not always exist.	false
Stop test if fails	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the test is immediately stopped if the assertion fails.	false
Assertion comment	String	No	An optional comment for information.	



Contribute in GitHub: [Edit online](#)

# Assert Less

Asserts that the value of the element identified by a given expression is less than a specific value.

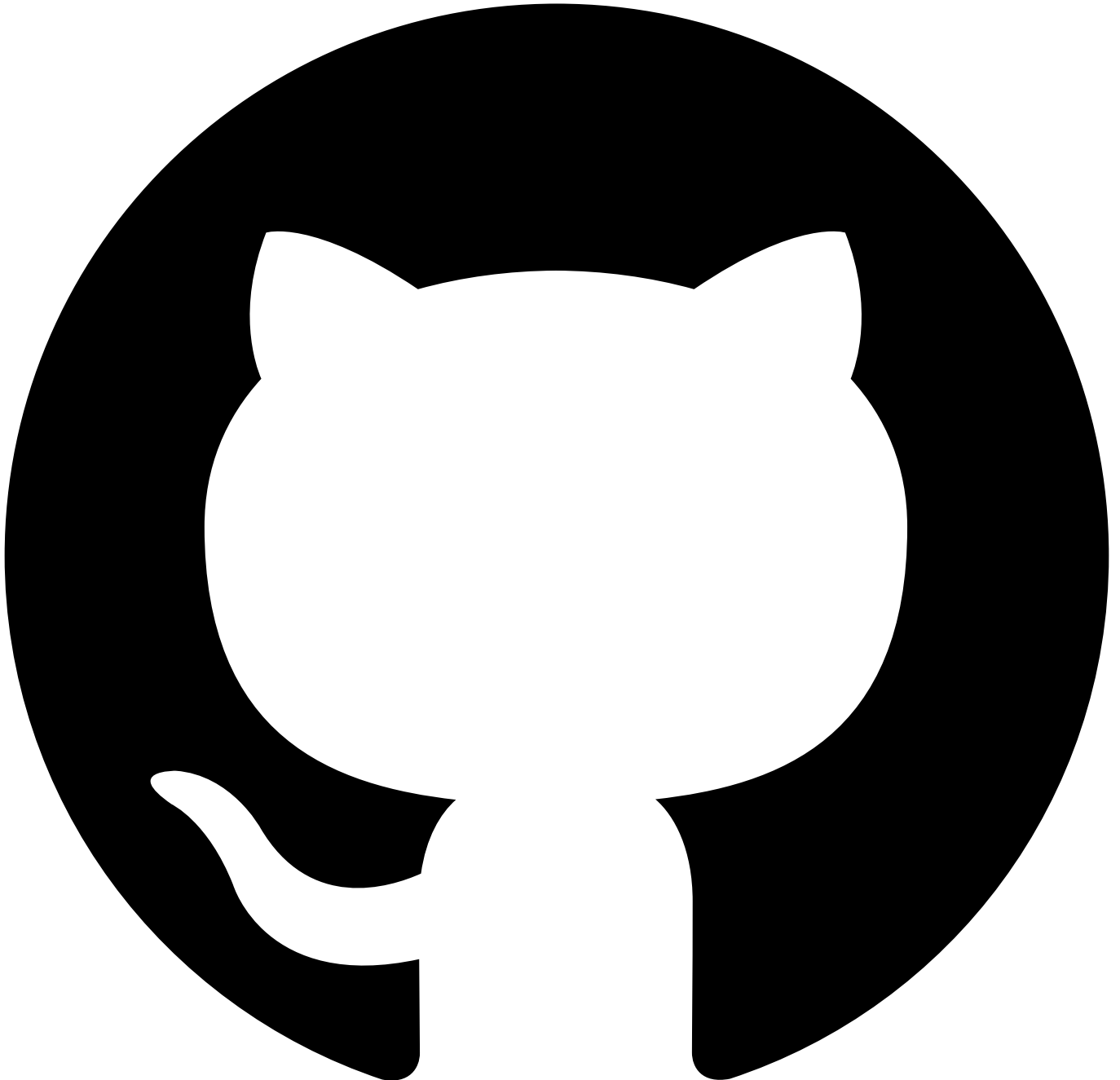
---

## Parameters

Name	Type	Required?	Description	Default value
Expression	Expression	Yes	Path to the element to be compared with the specified value; for example, <code>payload.productId</code> .	
Value	String	Yes	The value that the element is to be compared with.	
Type	Selection list	No	The data type of the value. If you do not specify a data type, the values are compared as strings. Select one of the following values: <ul style="list-style-type: none"><li><b>integer</b>: the values are evaluated as whole numbers.</li><li><b>float</b>: the values are evaluated as decimal numbers.</li></ul>	



Name	Type	Required?	Description	Default value
Mode	Selection list	No	Select one of the following values: <ul style="list-style-type: none"> <li><b>all:</b> If the path expression matches multiple elements in the payload then they must all match the assertion.</li> <li><b>one:</b> Only one of the matching elements in the payload needs to match the assertion.</li> </ul>	all
Level	Selection list	No	Select <b>error</b> or <b>warning</b> . Specifies whether, if the assertion fails, it should be considered an 'error' or just a 'warning.' A warning does not trigger alerts, such as email or text messages.	error
Modifier	Selection list	No	Select <b>not</b> to negate the assertion; the assertion is considered verified if it does not pass.	None
Execute if item exists	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the assertion is evaluated only if the element exists. This is useful when the element does not always exist.	false
Stop test if fails	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the test is immediately stopped if the assertion fails.	false
Assertion comment	String	No	An optional comment for information.	



Contribute in GitHub: [Edit online](#)

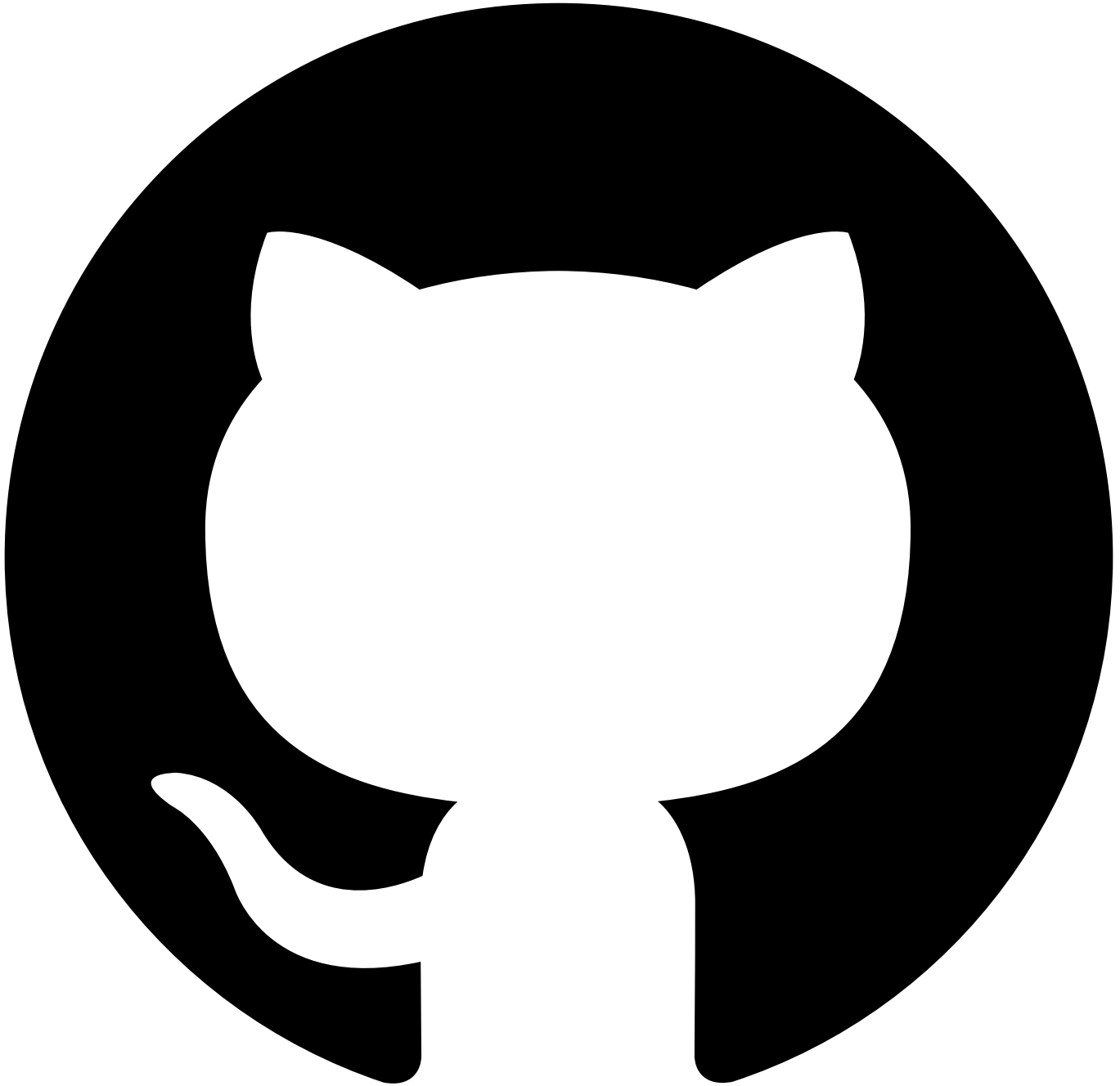
---

## Assert Matches

Asserts that the value of the element identified by a given expression matches a specified format.

## Parameters

Name	Type	Required?	Description	Default value
Expression	Expression	Yes	Path to the element that you want to examine; for example, <code>payload.productId</code> .	
Type	Selection list	Yes	<ul style="list-style-type: none"> <li><b>regex</b>: Evaluate the element value as a regular expression, as specified in the <b>Regex value</b> parameter.</li> <li><b>US states</b>: Check if the element value is a valid US State; for example, 'NY'.</li> <li><b>US zipcodes</b>: Check if the element value is a valid US Zipcode.</li> <li><b>credit card</b>: Check if the element value contains a valid credit card number from the most popular credit cards.</li> <li><b>country codes</b>: Checks if the element value contains a valid country code.</li> <li><b>currency codes</b>: Checks if the element value is a valid currency code.</li> </ul>	
Mode	Selection list	No	Select one of the following values: <ul style="list-style-type: none"> <li><b>all</b>: If the path expression matches multiple elements in the payload then they must all match the assertion.</li> <li><b>one</b>: Only one of the matching elements in the payload needs to match the assertion.</li> </ul>	all
Level	Selection list	No	Select <b>error</b> or <b>warning</b> . Specifies whether, if the assertion fails, it should be considered an 'error' or just a 'warning.' A warning does not trigger alerts, such as email or text messages.	error
Modifier	Selection list	No	Select <b>not</b> to negate the assertion; the assertion is considered verified if it does not pass.	None
Execute if item exists	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the assertion is evaluated only if the element exists. This is useful when the element does not always exist.	false
Stop test if fails	Selection list	No	Select <b>true</b> or <b>false</b> . If <b>true</b> , the test is immediately stopped if the assertion fails.	false
Assertion comment	String	No	An optional comment for information.	



Contribute in GitHub: [Edit online](#)

---

## Set Variable

**Set Variable** is a component that can be added to the test to capture a particular value for later use or to perform computations to arrive at a given result.

---

## Parameters

Name	Type	Required?	Description	Default value
Variable Name	String	Yes	Name of variable to be used in subsequent components	
Mode	Selection List	Yes	Select one of the following values: <ul style="list-style-type: none"><li>• <b>String:</b> Provide a string for the value of the variable, this can be referencing a payload property</li><li>• <b>Object:</b> Interrogate the object being processed</li><li>• <b>Language:</b> Using Groovy script provide logic to determine a value for the variable</li></ul>	String
Value	String	No	Provide the Value of the variable if <b>String</b> was selected for <b>Mode</b>	

Name	Type	Required?	Description	Default value
Object	String	No	Provide the logic for use when <b>Mode</b> is set to <b>Object</b>	
Language	Selection List	No	The language you wish to script in, Groovy is the only option available	Groovy
Content	String	No	Your Groovy script logic	

## String

---

To set a variable from a requests response if one is given.

for example to set a variable to the **name** property Add **Set Variable Component from list** Provide the name of the variable Set the value to `${<request variable name>.name}`

If the variable being referenced is within a loop the variable name will be `_1`

## Setting a Variable using Groovy script

---

To set variable to a value that requires a bit of additional logic, you can write a segment of groovy script to process inputs and provide an single output.

To write script, within **Set Variable** change the **Variable Mode** to **Language**

As an example if given a temperature I want to check if where it falls on a set of parameters using the tempature from a prior request

```
def temperature="Just Right"

if(payload.temperature>22){
 temperature='Its Warm'
} else if(payload.temperature<16){
 temperature='Its Cold'
}

return temperature
```

You much return a inner variable at the end to set the value of variable to be used later

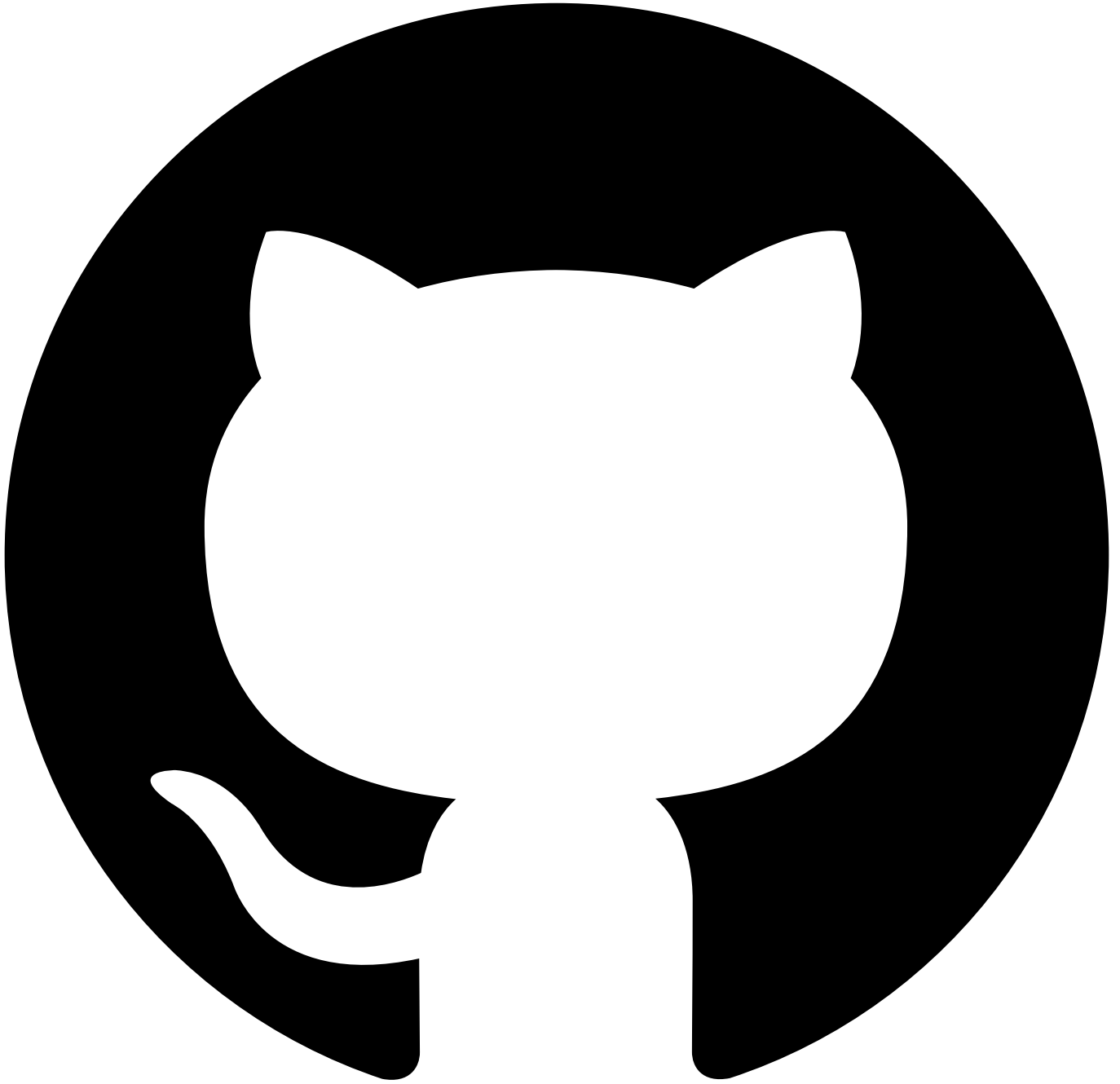
NOTE: within Groovy script you do not need to wrap existing variables within `${ }`

## Object Mode

---

Object allows you to inspect the a larger object to find particular parts of a varialbe

```
payload.findAll {it.name == 'Douglas Adams' }
```



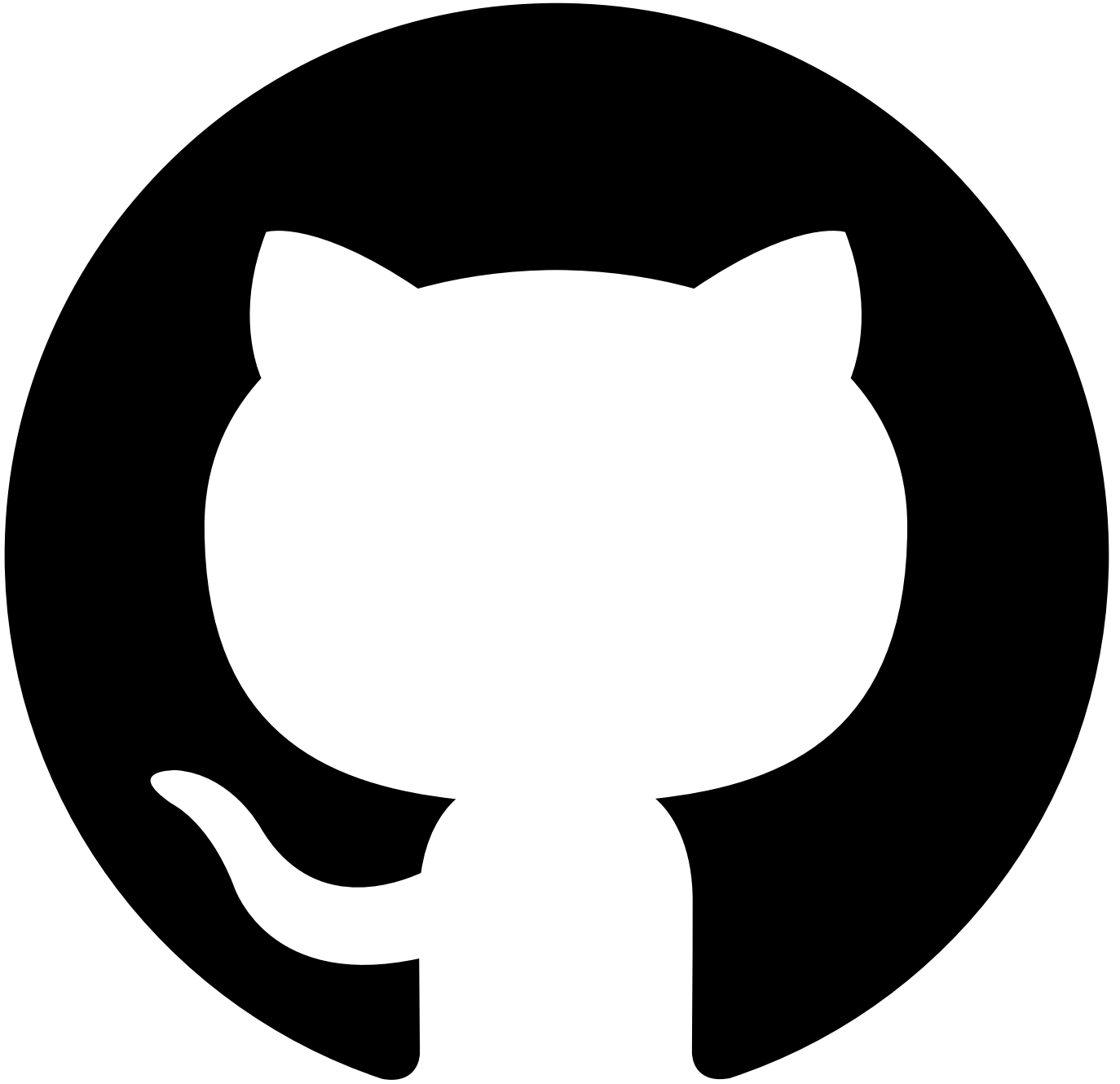
Contribute in GitHub: [Edit online](#)

---

## Security

It is important to be aware of potential security implications when using IBM API Connect Test APIs.

- [Storing sensitive data](#)



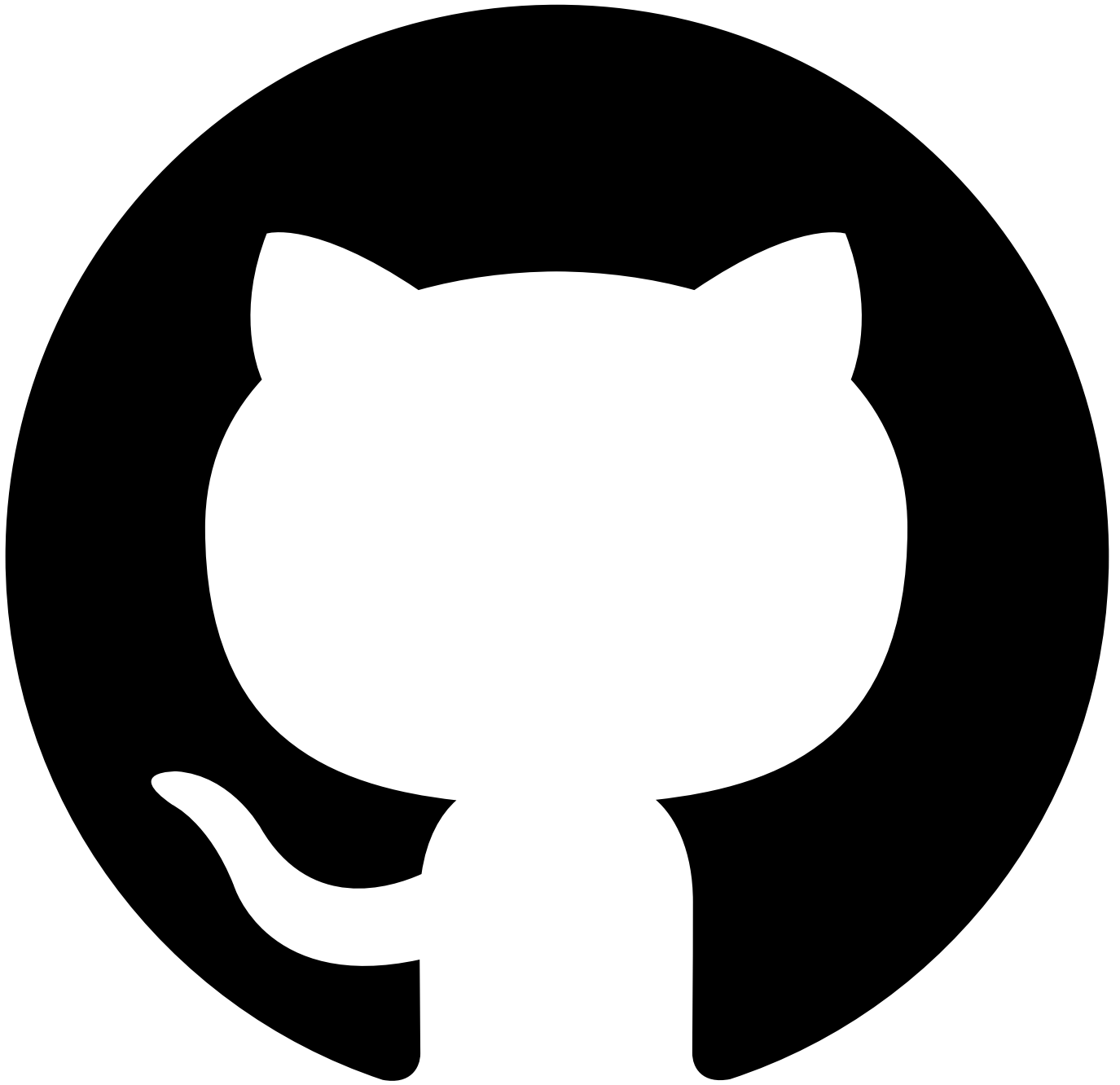
Contribute in GitHub: [Edit online](#)

---

## Storing sensitive data

When using IBM API Connect Test APIs, it is always helpful to remember:

- Before making a request to any endpoint, it is important to pause and think about the data supplied in the request and the data expected in the response. If the data contains sensitive data such as authentication credentials or any form of personal data, then we **strongly** discourage you from making that request.
- Try to always use test accounts when monitoring production services.



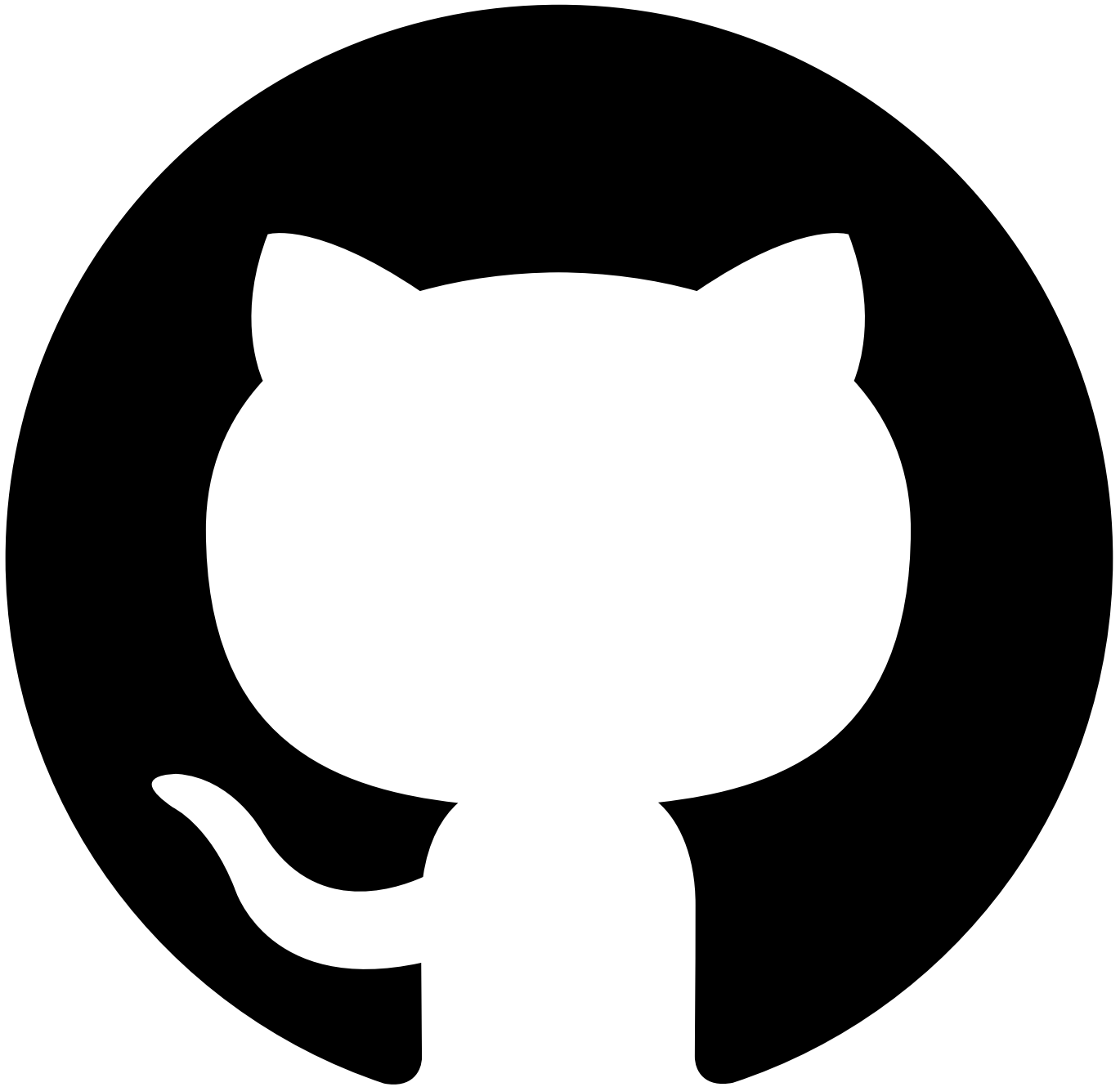
Contribute in GitHub: [Edit online](#)

---

## Additional tasks

Additional tasks with Test APIs.

- [Deleting a test suite](#)
- [Deleting a test](#)
- [Opening a test in the Editor](#)



Contribute in GitHub: [Edit online](#)

---

## Template Test from Specification Document

As an alternative to making requests against an existing API deployment to create a test, instead you can create a test template from a document that specifies the APIs behavior, or use a Postman Collection.

The template will include parameterized requests with payloads if they can be determined and assertions based on the API document.

Once the template has been generated, you may need to make changes to the datasets to use the correct endpoint and complete any request payloads that have been partially generated.

---

## Supported document types

Test Templating supports generating test cases from the following document types

- Swagger 2.0 JSON
- Swagger 2.0 YAML
- SOAP WSDL (single wsdl file)
- Postman Collection



## Templating Options

---

Once the document has been uploaded, you will be presented with a list of requests based on Path and expected status code

### **From Scratch**

Generates a single request and any assertions that can be determined about that request

This replaces an existing test definition

### **Merge**

Generate additional request and adds to the end of the existing test case

Note: This does not include any assertions, it is recommended the core request to be tested is generated from scratch and then additional requests merged into it

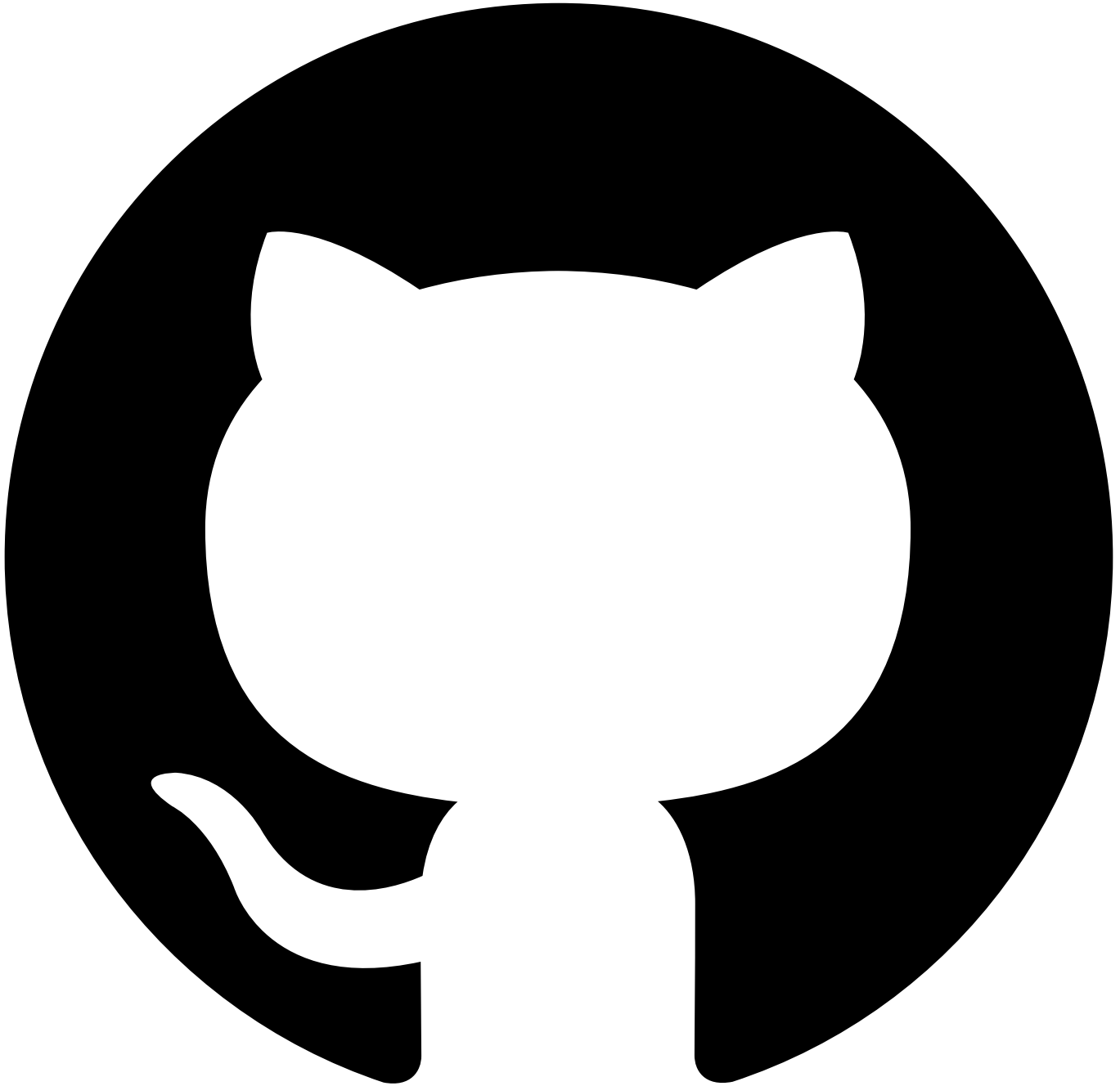
### **Smart Generation**

Restrictions:

- OpenAPI 2.0/Swagger 2.0 only
- Enterprise as a Service and Cloud Pak for Integration only

Once you've enabled the insights-driven test generation feature you will be able to generate a test template which can contain multiple related requests that include setup and teardown operations for the targeted operation.

Smart Generation uses static analysis and natural language processing of the API document to determine links between API requests and data that can be shared between the requests



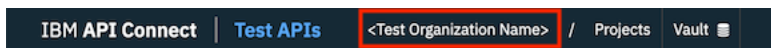
Contribute in GitHub: [Edit online](#)

---

## Deleting a test suite

To delete a test suite, complete the following steps:

1. Click your test organization name in the menu bar to open your test organization test suite page.



2. On the test suites page, locate the test suite you want to delete and determine whether it contains any tests by observing the displayed **Tests** count.



**Note:** You can't delete a test suite that contains tests, so if your test suite does contain any tests you must delete them first; see [Deleting a test](#).

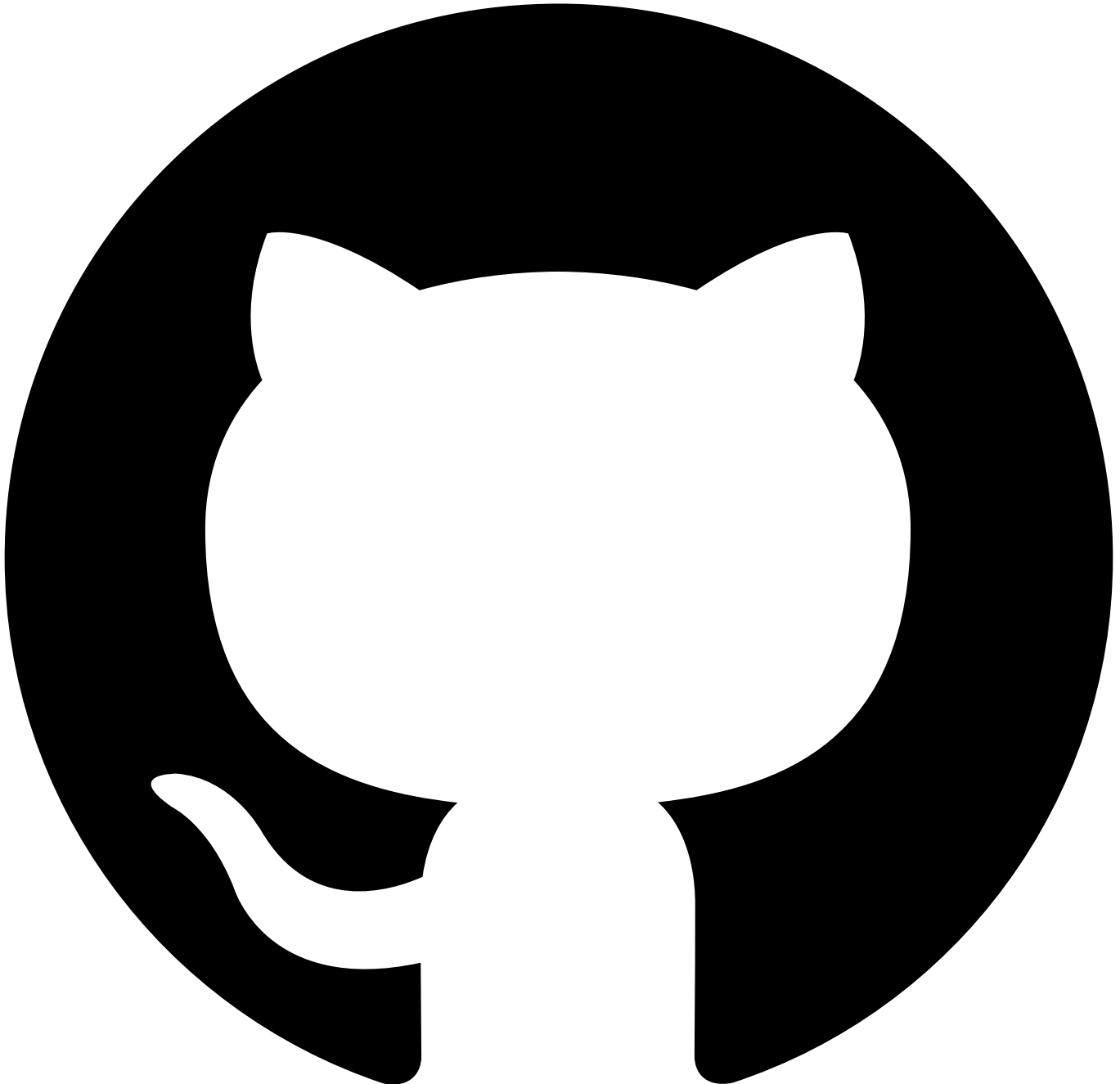


3. To delete the test suite, click its **Settings** icon [Settings](#), then click **Remove Test Suite**

**Note:** If the test suite contains tests, you will get an error when you click the **Settings** icon.

!Image of delete error](./dist/images/cant-delete-test suite.png)

If your test suite is deleted successfully, you are returned to your test organization test suite page, and your deleted test suite is no longer displayed.



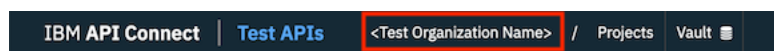
Contribute in GitHub: [Edit online](#)

---

## Deleting a test


To delete a test, complete the following steps:

1. Click your test organization name in the menu bar to open your test organization test suite page.




2. On the test suites page, locate the test suite you want to work with and ensure that it contains tests by confirming that the displayed **Tests** count is not zero.

My-project

Tests	1	
Events	0	
Failures	0	

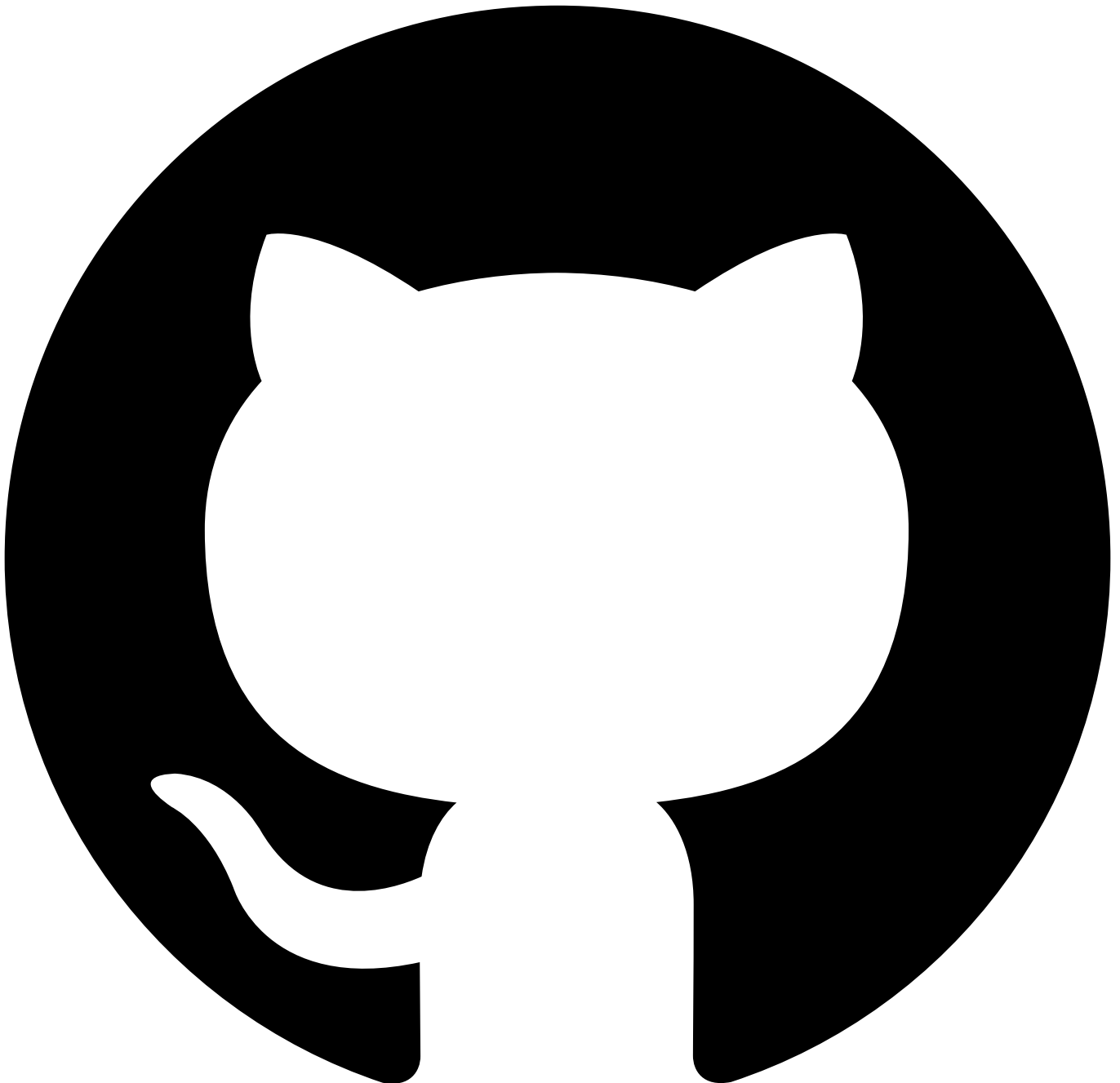


3. Click the test suite's **Tests** icon **Tests** . A list of the tests that have been created in the test suite are displayed.

4. Double-click the test you want to work with, or click its edit icon  .



5. On the test details page, click the **Delete Test** icon **Delete Test**, then click the tick icon to confirm. You are returned to the tests list and your deleted test is no longer displayed.



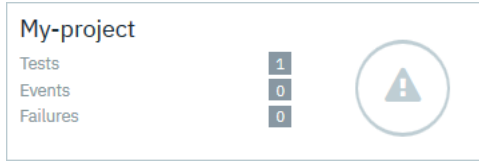
## Opening a test in the Editor


To open a test in the Editor, complete the following steps:


1. Click your test organization name in the menu bar to open your test organization test suite page.



2. On the test suites page, locate the test suite you want to work with and ensure that it contains tests by confirming that the displayed **Tests** count is not zero.

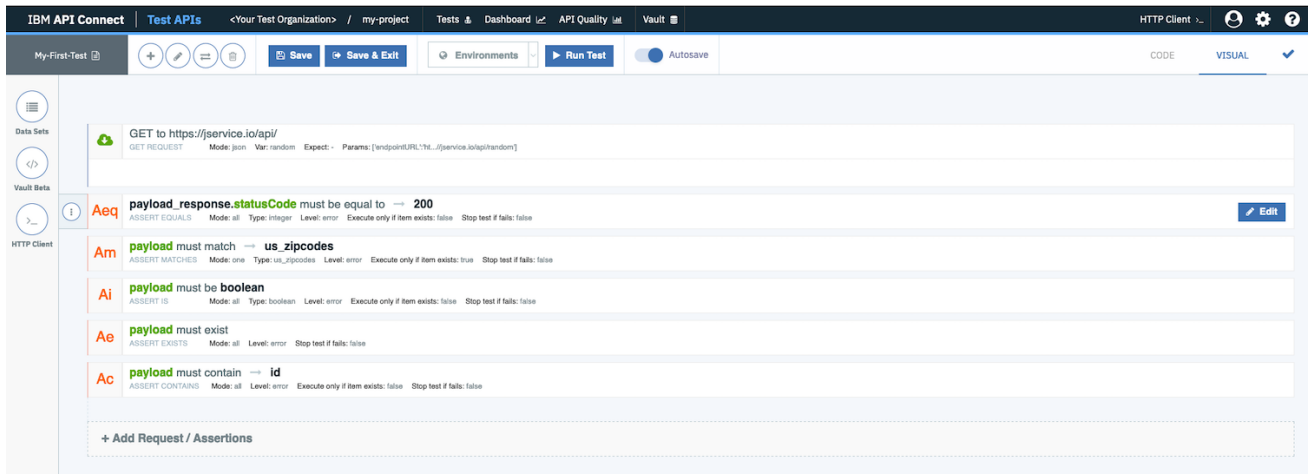


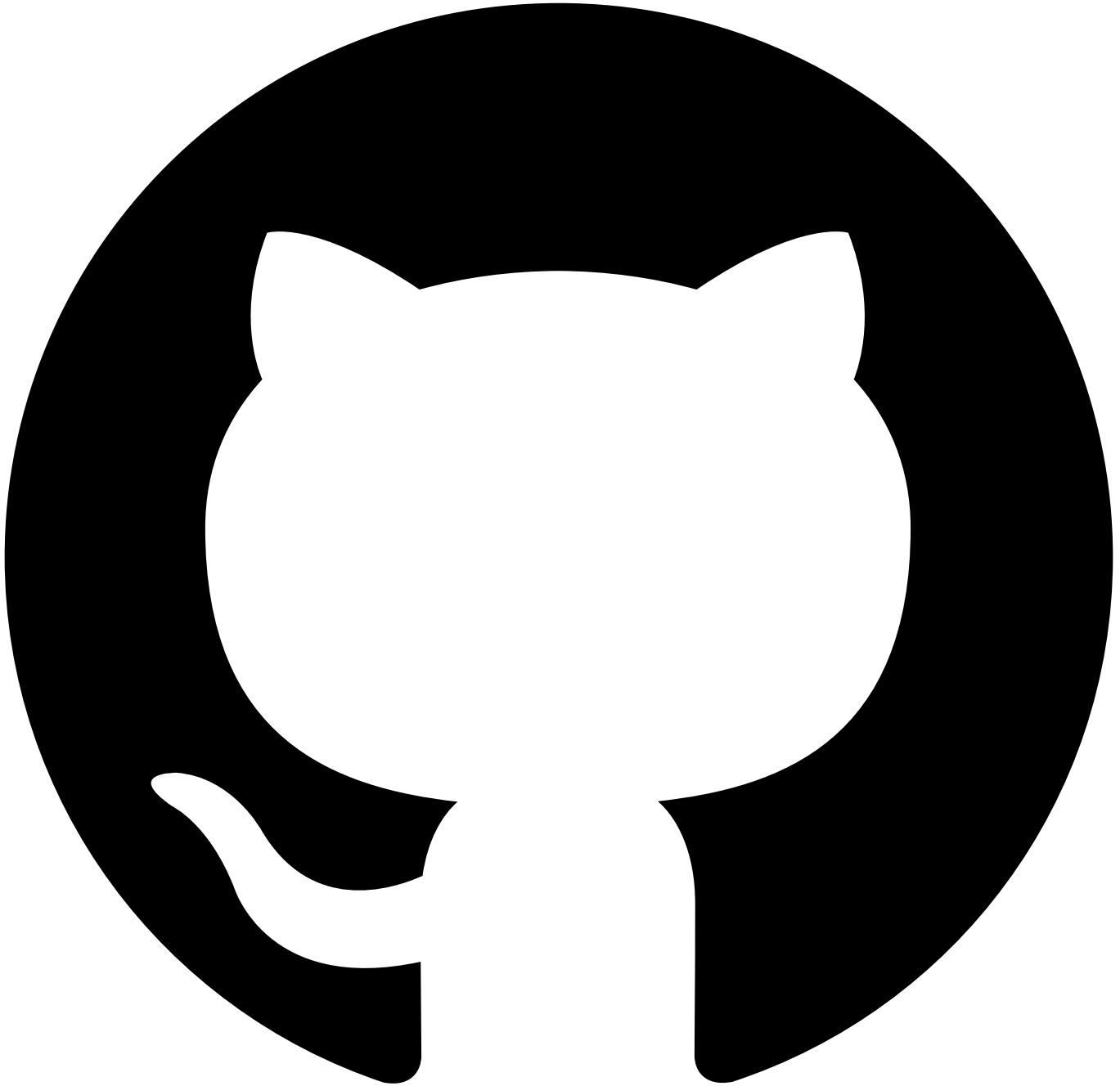
3. Click the test suite's **Tests** icon . A list of the tests that have been created in the test suite are displayed.

4. Double-click the test you want to work with, or click its edit icon .



5. On the test details page, click the **Compose** icon . The Editor page opens for the selected test; for example:





Contribute in GitHub: [Edit online](#)

---

## API Hooks

In addition to viewing and executing tests within the IBM API Connect Test interface, you may also perform these actions via an API. This allows tests to be executed via the command line, through third-party applications, or as part of a CI/CD pipeline. API hooks form the basis of this API, providing a base URL on which subsequent API operations can be performed. To start using an API Hook, see:

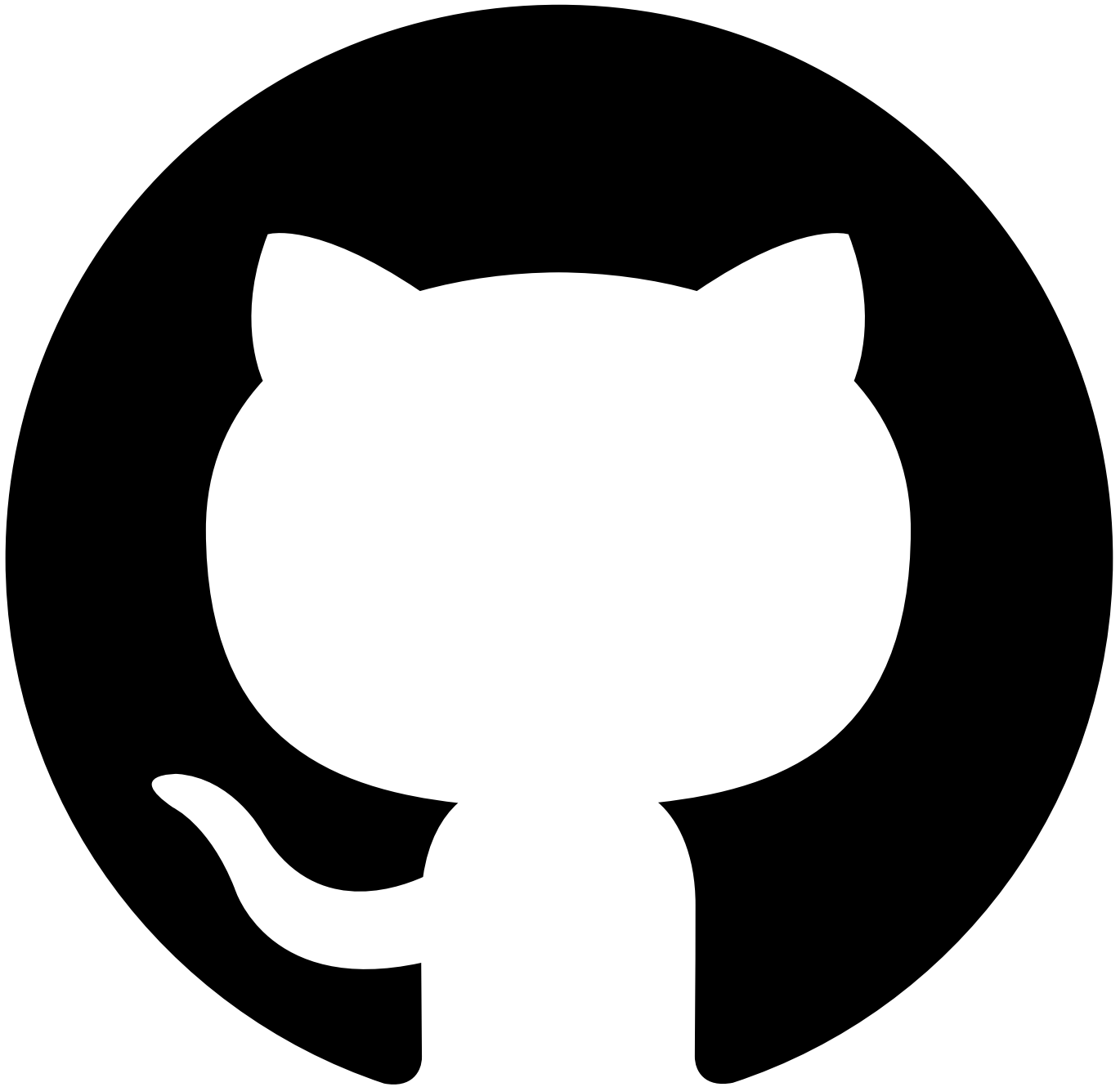
- [Creating An API Hook](#).

API Hooks require authentication to use in the form of an API Key and Secret. Once you have created your API Hook, view the following page to learn how to create an API Key and Secret for your Test Organization:

- [Creating API Keys and Secrets](#)

Once you've set up your API Hook, Key and Secret, see:

- [Using API Hooks](#)



Contribute in GitHub: [Edit online](#)

---

## Creating an API Hook

You can generate an API hook for each of the Test Suites within your Test Organization. API hooks are scoped to only one test suite and thus will only have access to operate on the data within that test suite; that is, if an API Hook is generated for **Test Suite 1**, it may only access the tests and test results associated with **Test Suite 1**. If you wish to interact with tests in multiple test suites, you will need to generate an API hook for each Test Suite.

API Hooks are protected and require an API Key and Secret to be used. You can find out how to create a API Key and Secret [here](#).

---

## Prerequisites

1. Your account must have administrators or Test Organization manager privileges.
2. You must have at least one Test Suite in your Test Organization.

To see a list of all the Test Suites in your Test Organization, navigate to the Main Dashboard by clicking on your Test Organization Name in the menu bar.



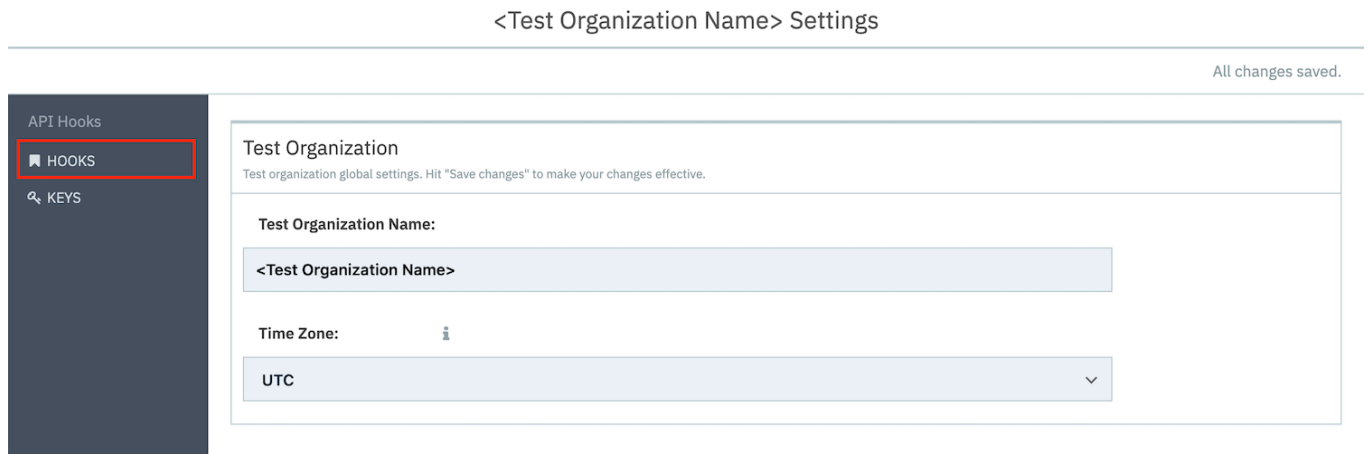
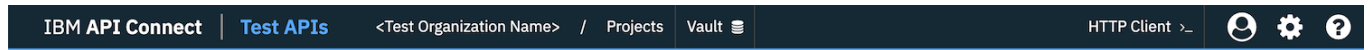
If there are no Test Suites here, you can create one by clicking the **Create Test Suite** button after the Test Status chart.

## How to create an API Hook

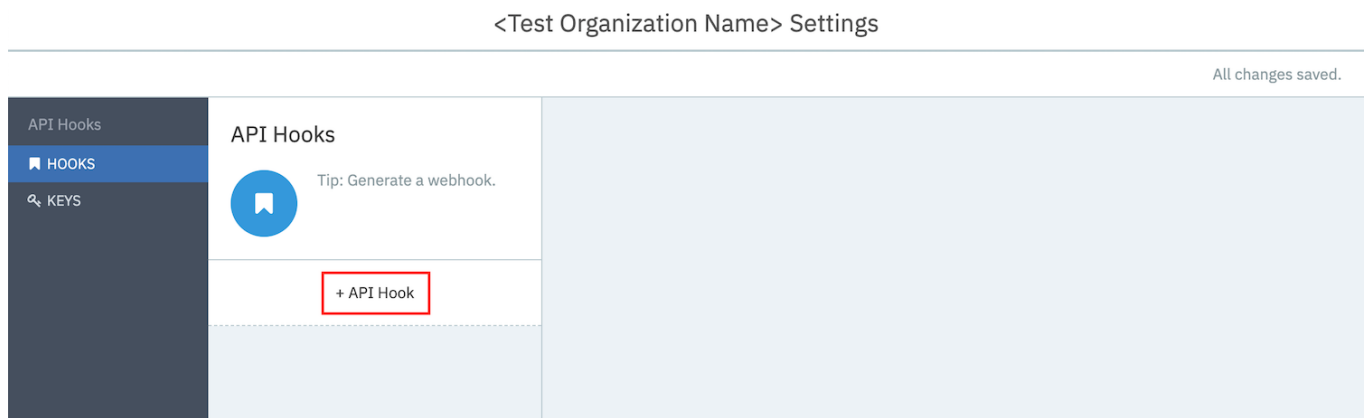
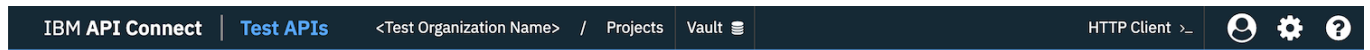
First navigate to the Test Organization settings page by clicking the **Cog** icon in the menu bar.




Within the Test Organization settings page, click the **Hooks** option under the API Hooks heading in the navigation menu.



Now click on the **+ API Hook** button to begin creating a new API Hook.



As API Hooks are scoped at a Test Suite level, you will need to select the test suite you wish to generate the hook for. Choose the appropriate test suite from the Test Suite dropdown and optionally add a description for the API Hook. Click the **confirm**  icon to generate and save the API Hook.

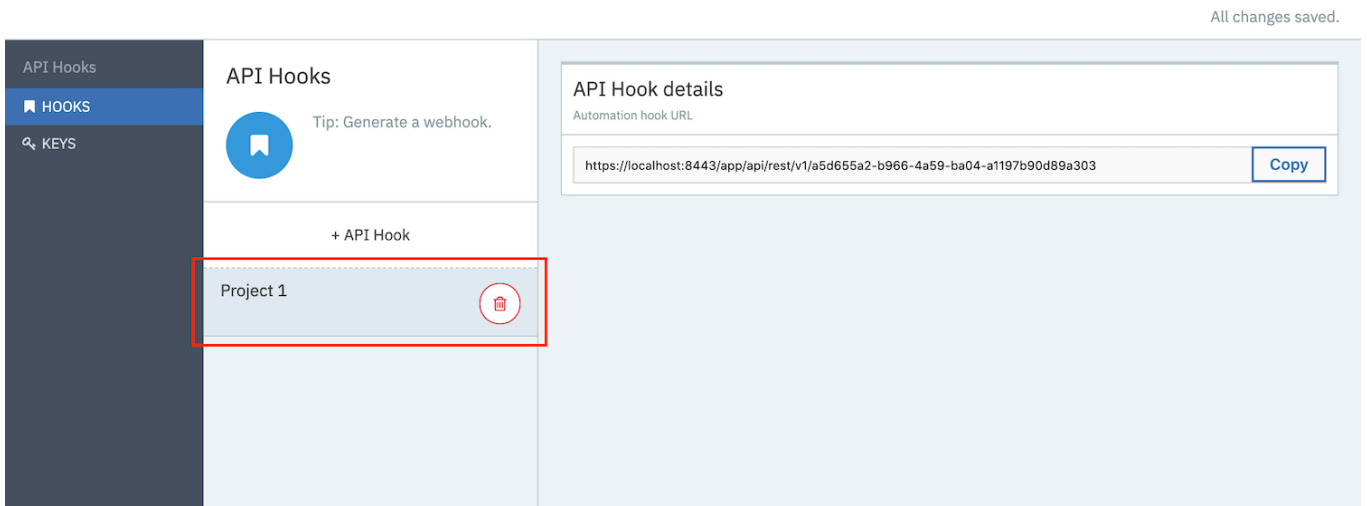
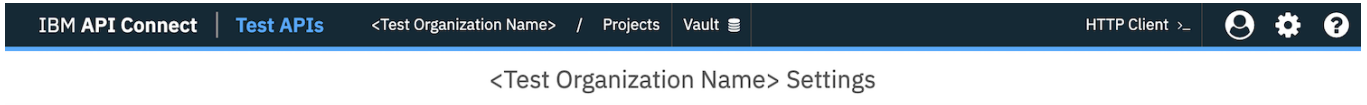


You've now created the API Hook URL for your Test Suite. Make a note of this URL as this will form the base path of all future API requests you make to this test suite.





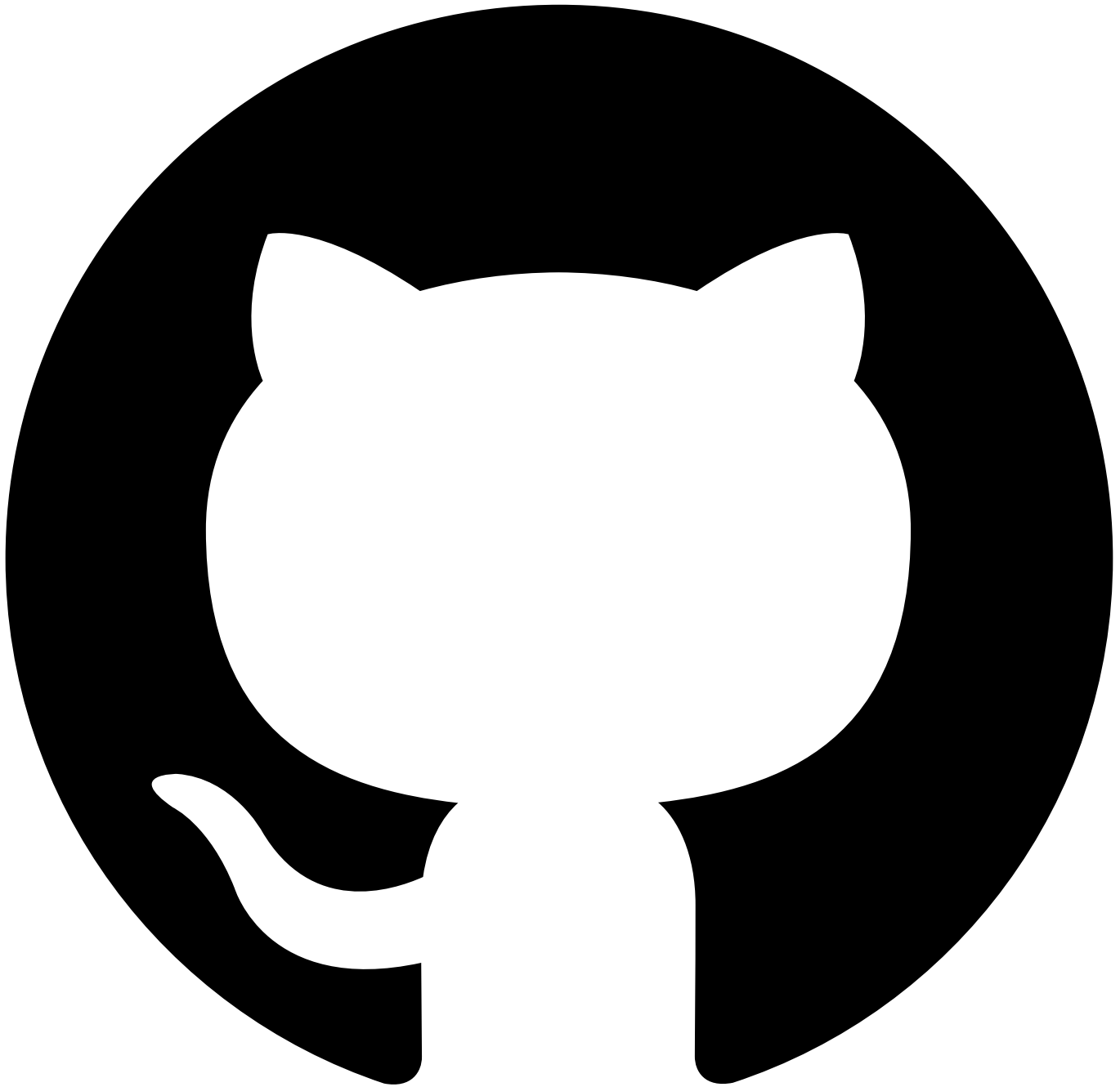
Close the Hook URL dialog box. You can see that your new API Hook is now in the list of API Hooks. You can click on this at anytime to view the URL again in the future should you need. Note that this URL will remain active until it is deleted. To delete the API Hook, simply click on the **Delete** icon next to the API Hook you want to remove.



In order to use your new API Hook, you need to ensure you have an API Key and Secret for your Test Organization. Steps do do this can be found [here](#). Alternatively, if your Key and Secret is set up, see [Using API Hooks](#) for more information on how to use the API Hook.

## What to do next

- Next Topic: [Creating API Keys and Secrets](#)



Contribute in GitHub: [Edit online](#)

---

## Creating API Keys and Secrets

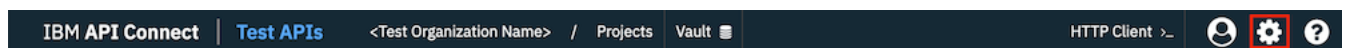
API Keys and Secrets are the credentials required to use an [API Hook](#). For a request to an API Hook to be authorised, both the **X-API-Key** and **X-API-Secret** headers must be provided. The values of the API Key and Secret represent the values of these headers respectively.

Unlike API Hooks which are scoped to a single Test Suite, API Keys and Secrets are scoped to a Test Organization. Therefore, any API Key and Secret pair which is generated within a Test Organization will have permission to authorise requests to all API Hooks within the same organization, regardless of the Test Suite the API Hook is associated with. API Keys and Secrets can only be generated by system administrators or test organization managers.

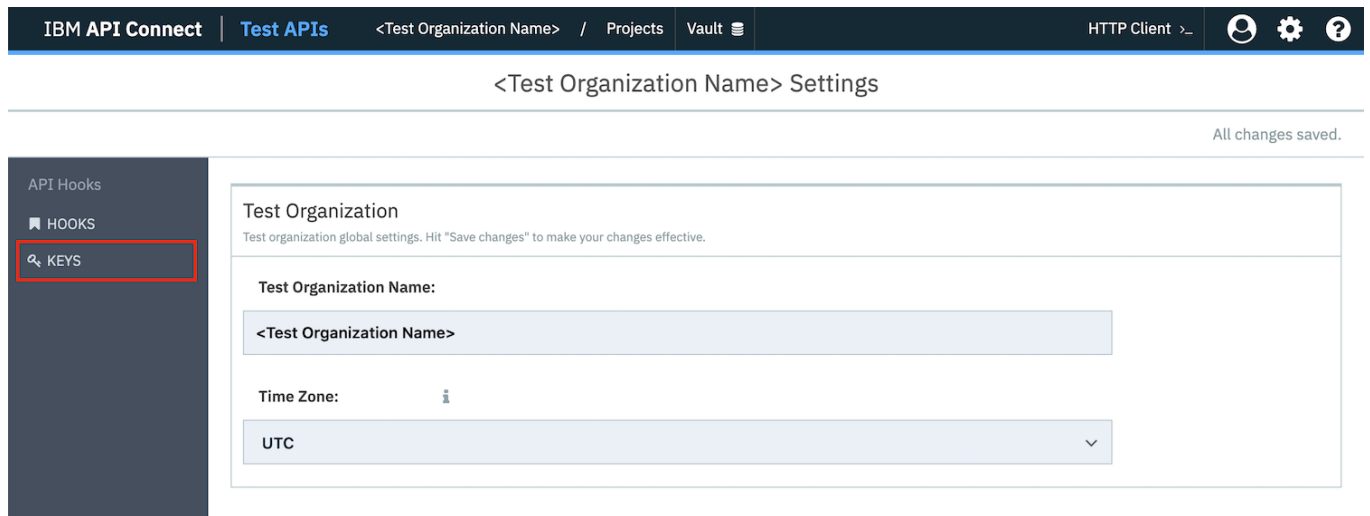
---

## Creating an API Key and Secret

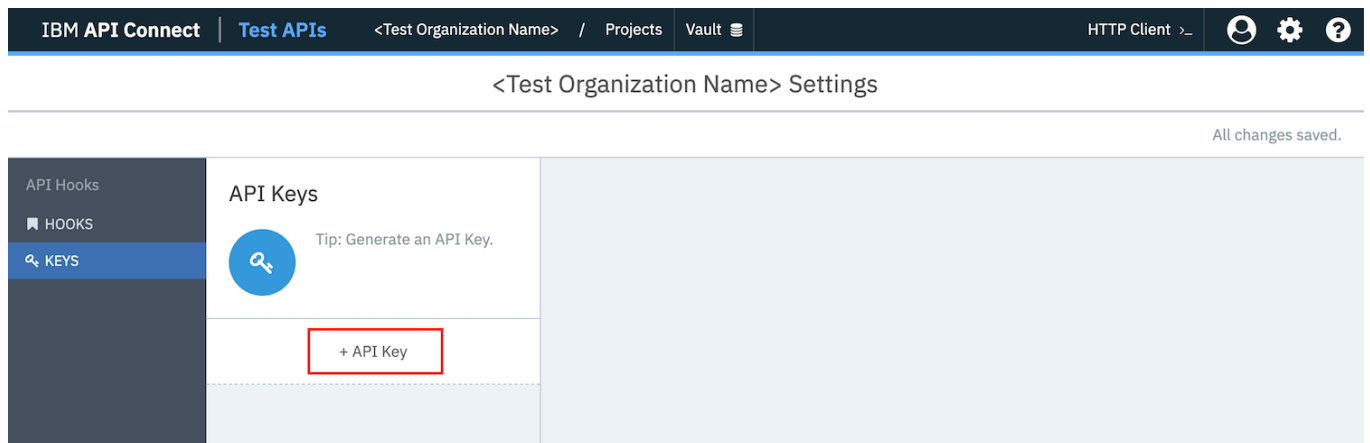
First navigate to the Test Organization settings page by clicking the **Cog** icon in the menu bar. Note that you will only see this icon if you are a administrator or a test organization manager.




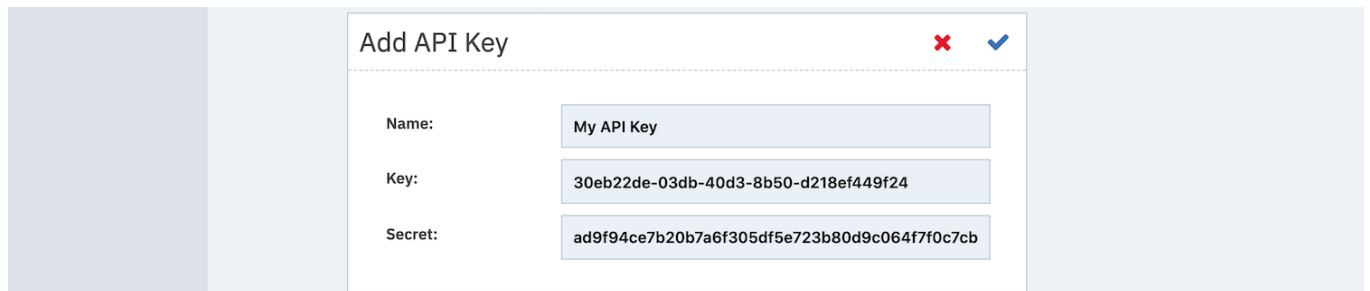
Within the Test Organization settings page, click the **KEYS** option under the API Hooks heading in the navigation menu.



Now click on the **+ API Key** button to create a new API Key.



You will then be shown the generated **Key** and **Secret**. Make a note of both of these, especially the **Secret** as this will be the only time this is visible. Give your new API Key a name and then click the **confirm**  icon to save it.



Your new API Key is now active and will now appear in the list of API Keys in the navigation menu. Clicking on the key will show you the API Key's value should you need to reference this again in the future. If you want to remove the API Key at any point, simply click on the **Delete** icon next to the API Key you want to remove. This will invalidate all future requests to API Hooks within you Test Organization using this Key and Secret combination.

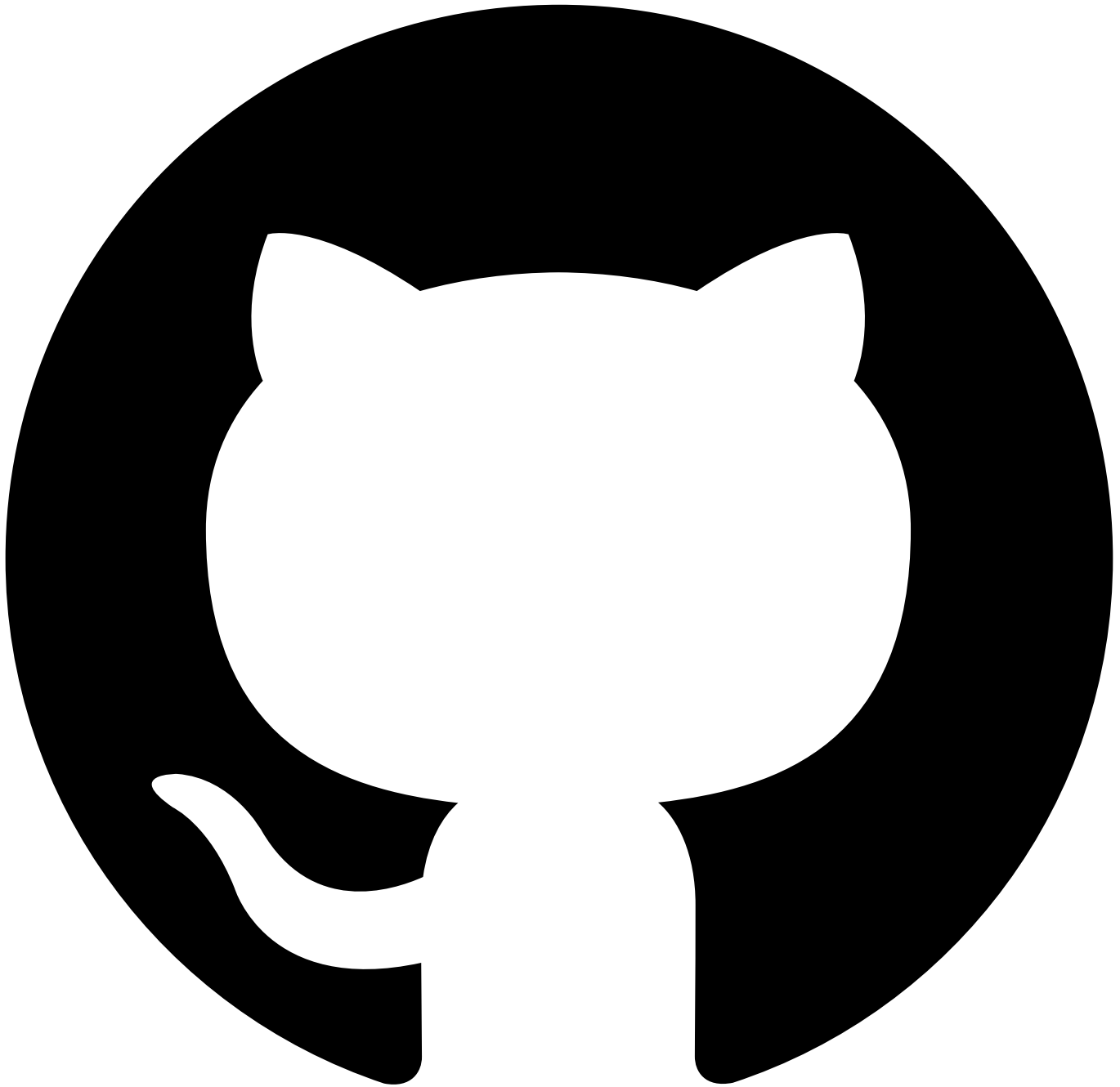
## <Test Organization Name> Settings

All changes saved.

The screenshot displays the 'API Keys' management interface. On the left, a sidebar contains 'API Hooks', 'HOOKS', and 'KEYS'. The main area is titled 'API Keys' and includes a tip: 'Tip: Generate an API Key.' Below this is a '+ API Key' button. A list of API keys is shown, with one entry 'My API Key' highlighted by a red box. The key's ID is '30eb22de-03db-40d3-8b50-d218ef449f24'. On the right, the 'API key details' panel shows the key value '30eb22de-03db-40d3-8b50-d218ef449f24' and a 'Copy' button.

## What to do next

- Next Topic: [Using API Hooks](#)



Contribute in GitHub: [Edit online](#)

---

## Using API Hooks

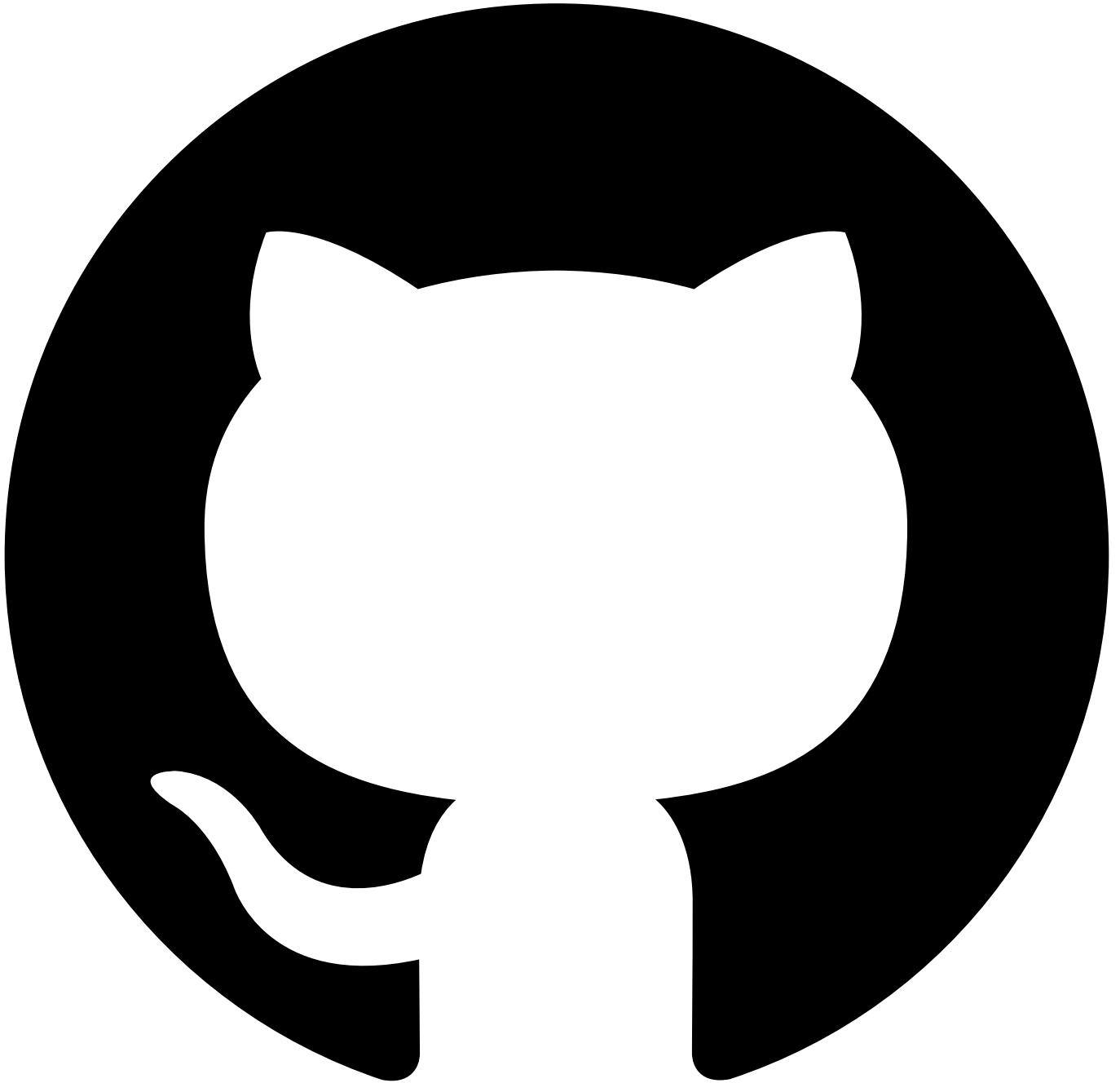
[API Hooks](#) provide a mechanism to interact with tests within a Test Suite via an API. Whilst the API Hook itself defines the basepath of this API, it also exposes a series of endpoints which, when appended onto this basepath, allow you to list and execute published tests within the API Hook's scope. The following pages detail these endpoints and subsequent parameters you can provide to either list, execute all, or execute only specific tests within a test suite.

---

## Endpoints

- [List All Tests](#)  
Lists all of the published tests within an API Hook's scope.
- [Run All Published Tests](#)  
Runs all of the published tests within an API Hook's scope.
- [Run Published Tests By Tag](#)  
Runs the published tests within an API Hook's scope that match the tags specified in the request body.

- [Run Published Test By ID](#)  
Run a specified test within an API Hook's scope.



Contribute in GitHub: [Edit online](#)

---

## List All Tests

### Description

---

The List All Tests endpoint lists all of the published tests within an API Hook's scope.

### HTTP Request

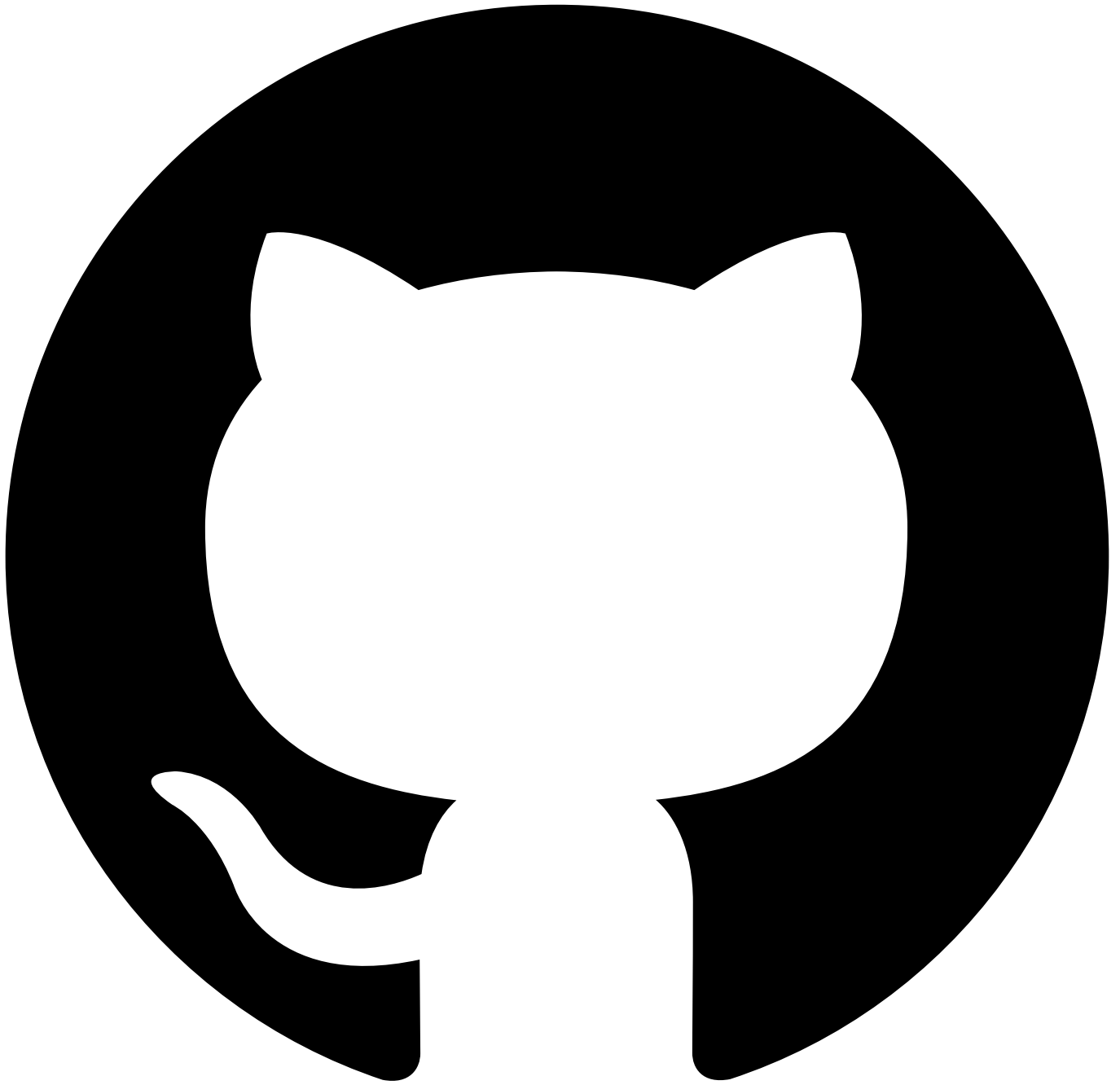
---

```
GET <API-Hook-URL>/tests
```

## Sample Method Invocation

```
curl -X GET \
-H X-API-Key:{API-Key} \
-H X-API-Secret:{API-Secret} \
<API-Hook-URL>/tests
```

<b>Parameters</b>	<b>Header Parameters</b> API-Key: A valid <a href="#">API Key</a> for the Test Organization this API Hook belongs to. API-Secret: A valid <a href="#">API Secret</a> for the Test Organization this API Hook belongs to.
<b>Response Content</b>	<b>MIME type: application/json</b> <pre>[   {     testId: string,     testName: string,     testDescription: string,     lastModified: string,     authorName: string,     tags: array[string]   } ]</pre> <b>Response Description</b> <ul style="list-style-type: none"><li>• <b>testId</b><ul style="list-style-type: none"><li>◦ ID of the test.</li></ul></li><li>• <b>testName</b><ul style="list-style-type: none"><li>◦ Name of the test.</li></ul></li><li>• <b>testDescription</b><ul style="list-style-type: none"><li>◦ Description of the test.</li></ul></li><li>• <b>lastModified</b><ul style="list-style-type: none"><li>◦ Date and time the test was last modified.</li></ul></li><li>• <b>authorName</b><ul style="list-style-type: none"><li>◦ User who composed the test.</li></ul></li><li>• <b>tags</b><ul style="list-style-type: none"><li>◦ Array of tags associated with the test.</li></ul></li></ul>
<b>Status Codes</b>	200: Success 400: Missing Authorization Headers 401: Invalid Authorization Headers



Contribute in GitHub: [Edit online](#)

---

## Run All Published Tests

### Description

---

The Run All Tests endpoint executes all of the published tests within an API Hook's scope.

### HTTP Request

---

```
POST <API-Hook-URL>/tests/run
```

---

## Sample Method Invocation



```

curl -X POST \
-H X-API-Key:{API-Key} \
-H X-API-Secret:{API-Secret} \
-H Content-Type:application/json \
-d "
 {
 "options": {
 "allAssertions": boolean,
 "JUnitFormat": boolean
 },
 "variables": {
 string: string,
 }
 }
" \
<API-Hook-URL>/tests/run

```

<b>Parameters</b>	<b>Header Parameters</b>  API-Key: A valid <a href="#">API Key</a> for the Test Organization this API Hook belongs to. API-Secret: A valid <a href="#">API Secret</a> for the Test Organization this API Hook belongs to.
<b>Request Body</b>	<b>MIME type: application/json</b>  <pre> {   options: {     allAssertions: boolean,     JUnitFormat: boolean   },   variables: {     string: string,   } } </pre> <b>Body Parameters</b> <ul style="list-style-type: none"> <li>• <b>options</b> (Only required if providing a child element) <ul style="list-style-type: none"> <li>◦ A wrapper object for the containing options on how what data should be returned in the response, or how the response should be formatted.</li> <li>◦ Child Elements: <ul style="list-style-type: none"> <li>▪ <b>allAssertions</b> (Optional) <ul style="list-style-type: none"> <li>▪ When <b>false</b>, only the results of failing assertions in the test are returned. When <b>true</b>, all assertions in the test are returned. <b>allAssertions</b> is <b>false</b> by default.</li> </ul> </li> <li>▪ <b>JUnitFormat</b> (Optional) <ul style="list-style-type: none"> <li>▪ When <b>false</b>, <b>application/json</b> format is returned for the test results. When <b>true</b>, <b>text/xml</b> JUnit format is used for the test results (see <b>text/xml</b> response). <b>JUnitFormat</b> is <b>false</b> by default.</li> </ul> </li> </ul> </li> </ul> </li> <li>• <b>variables</b> (Optional) <ul style="list-style-type: none"> <li>◦ The variables section is used to pass a list of named variables and their values into the tests being run. It is only used by tests that have been written to make use of variables, and have values stored alongside them. Declaring the variable name as the object key, and its new value as the key's value (as type string), will overwrite the current value of the variable in the test which shares its name with the object key.</li> </ul> </li> </ul>
<b>Response Content</b>	<b>MIME type: application/json</b>  <b>application/json</b> is the default response from the endpoint.  <pre> [   {     testRunId: string,     testId: string,     testName: string,     location: string,     initiated: string,     duration: integer,     status: enum['passed' or 'failed'],     reportUrl: string,     results: {       warningsCount: integer,       failureCount: integer,       httpFailures: array[object],       criticalFailures: array[object],       assertions: array[object]     }   } ] </pre>

## Response Description

- **testRunID**
  - ID of the test run.
- **testId**
  - ID of the test which was executed.
- **testName**
  - Name of the test.
- **location**
  - Location of the agent which executed the test.
- **initiated**
  - Date and time when the test was initiated.
- **duration**
  - Total duration of the test run in seconds.
- **status**
  - Overall result of the test.
- **reportUrl**
  - URL to the test run report.
- **results**
  - A wrapper object containing the results of the test run.
  - Child Elements:
    - **warningsCount**
      - Count of the HTTP warnings returned by the test assertions.
    - **failureCount**
      - Count of the HTTP status code responses that are greater than 400.
    - **httpFailures**
      - Description of failures whose HTTP Status response code is greater than or equal to 400.
    - **criticalFailures**
      - Description of failures whose HTTP Status response code is greater than or equal to 500.
    - **assertions**
      - A description of the assertions executed in the test. By default, only failing assertions will be listed.

## MIME type: text/xml

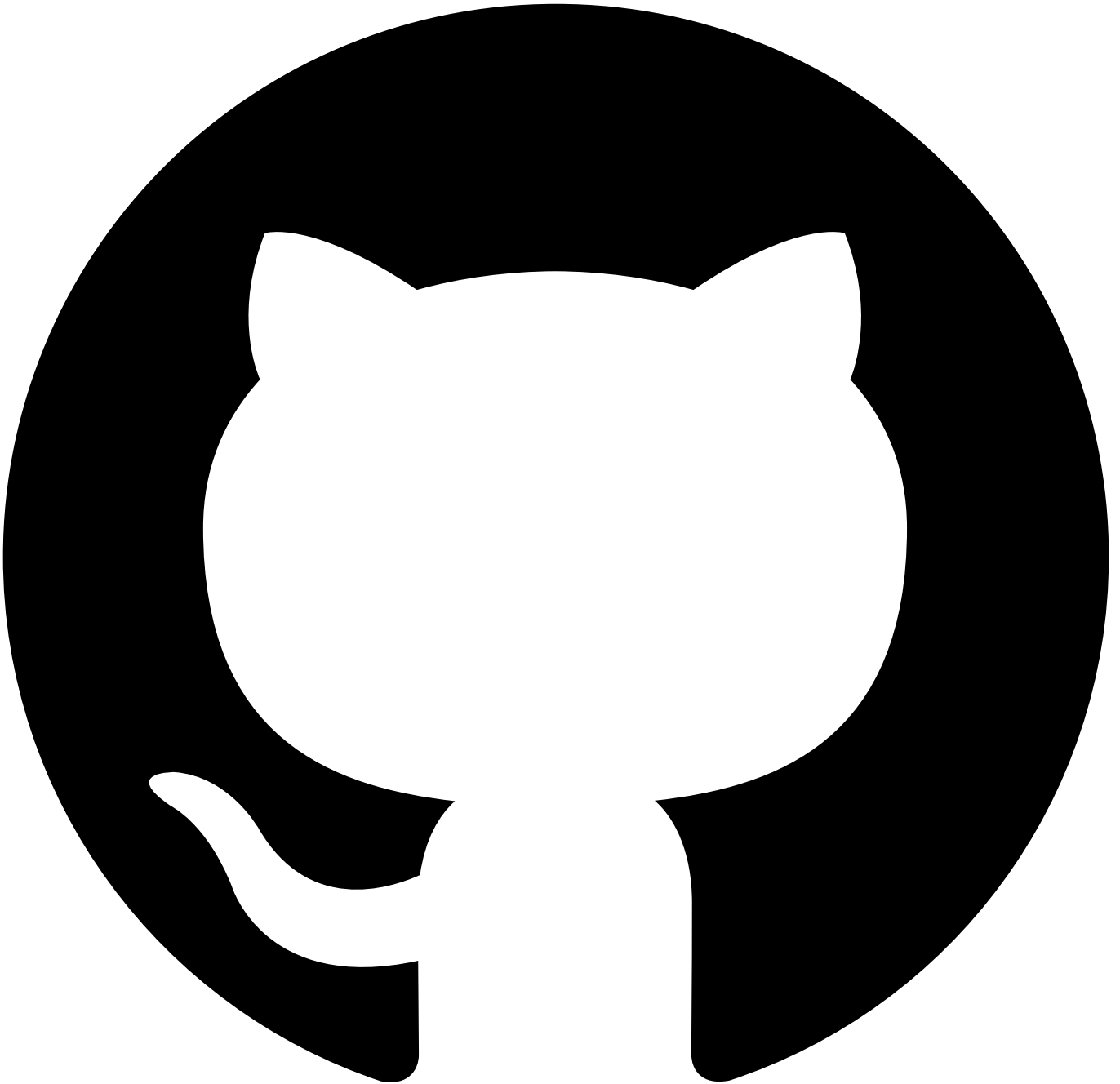
To receive a **text/xml** response, the optional **JUnitFormat** variable in the request body must be set to **true**. JUnit responses can be useful when you wish to incorporate your test results with a build pipeline such as Jenkins, via the [JUnit plugin](#).

```
<?xml version="1.0" encoding="UTF-8"?>
<testSuite name="" tests="" disabled="" errors="" failures="" timestamp="" time="">
<testcase name="" classname="" time="">
<failure message="" type="" />
</testcase>
</testSuite>
```

## Response Description

- **<testSuite>**
  - Child Elements: **<testCase>**
  - Attributes:
    - **name**
      - Test Suite name.
    - **tests**
      - Number of published tests ran.
    - **disabled**
      - Number of disabled tests.
    - **errors**
      - Number of errors.
    - **failures**
      - Number of tests within the test suite that failed.
    - **timestamp**
      - Date and time the tests were ran.
    - **time**
      - Total duration of all test runs in seconds.
- **<testCase>**
  - Child Elements: **<failure>**
  - Attributes:
    - **name**
      - Test name.
    - **classname**
      - ID of the test which was executed.
    - **time**
      - Total duration of the test run in seconds.
- **<failure>**

	<ul style="list-style-type: none"><li>◦ Attributes:<ul style="list-style-type: none"><li>▪ <b>message</b><ul style="list-style-type: none"><li>▪ Failure message containing the expected and actual response.</li></ul></li><li>▪ <b>type</b><ul style="list-style-type: none"><li>▪ Type of failure.</li></ul></li></ul></li></ul>
<b>Status Codes</b>	200: Success 400: Missing Authorization Headers 401: Invalid Authorization Headers



Contribute in GitHub: [Edit online](#)

---

## Run Published Tests By Tag

### Description

The Run Published Test By Tag endpoint executes all of the published tests within an API Hook's scope that match the tags specified in the request body. See [Tagging a Test](#) for more information on how to tag a test.

### HTTP Request

---

## Sample Method Invocation

```
curl -X POST \
-H X-API-Key:{API-Key} \
-H X-API-Secret:{API-Secret} \
-H Content-Type:application/json \
-d "
 {
 "options": {
 "allAssertions": boolean,
 "JUnitFormat": boolean
 },
 "variables": {
 string: string,
 },
 "tags": {
 "tagList": array[string],
 "operator": enum['not' or 'and' or 'or']
 }
 }
" \
<API-Hook-URL>/tests/run/tag
```

<b>Parameters</b>	<b>Header Parameters</b>  API-Key: A valid <a href="#">API Key</a> for the Test Organization this API Hook belongs to. API-Secret: A valid <a href="#">API Secret</a> for the Test Organization this API Hook belongs to.
<b>Request Body</b>	<b>MIME type: application/json</b>  <pre>{   options: {     allAssertions: boolean,     JUnitFormat: boolean   },   variables: {     string: string,   },   tags: {     tagList: array[string],     operator: enum['not' or 'and' or 'or']   } }</pre> <b>Body Parameters</b> <ul style="list-style-type: none"> <li>• <b>options</b> (Only required if providing a child element) <ul style="list-style-type: none"> <li>◦ A wrapper object for the containing options on how what data should be returned in the response, or how the response should be formatted.</li> <li>◦ Child Elements: <ul style="list-style-type: none"> <li>▪ <b>allAssertions</b> (Optional) <ul style="list-style-type: none"> <li>▪ When <b>false</b>, only the results of failing assertions in the test are returned. When <b>true</b>, all assertions in the test are returned. <b>allAssertions</b> is <b>false</b> by default.</li> </ul> </li> <li>▪ <b>JUnitFormat</b> (Optional) <ul style="list-style-type: none"> <li>▪ When <b>false</b>, <b>application/json</b> format is returned for the test results. When <b>true</b>, <b>text/xml</b> JUnit format is used for the test results (see <b>text/xml</b> response). <b>JUnitFormat</b> is <b>false</b> by default.</li> </ul> </li> </ul> </li> </ul> </li> <li>• <b>variables</b> (Optional) <ul style="list-style-type: none"> <li>◦ The variables section is used to pass a list of named variables and their values into the tests being run. It is only used by tests that have been written to make use of variables, and have values stored alongside them. Declaring the variable name as the object key, and its new value as the key's value (as type string), will overwrite the current value of the variable in the test which shares its name with the object key.</li> </ul> </li> <li>• <b>tags</b> (Required) <ul style="list-style-type: none"> <li>◦ A wrapper object containing both the list of tags and the operator to use on them.</li> <li>◦ Child Elements: <ul style="list-style-type: none"> <li>▪ <b>tagList</b> (Required) <ul style="list-style-type: none"> <li>▪ The list of tags which correspond to the tests you want to execute or ignore (see <b>operator</b> element).</li> </ul> </li> <li>▪ <b>operator</b> (Required) <ul style="list-style-type: none"> <li>▪ The operator indicates how to treat the list of tags and can be either one of three values: <b>not</b>, <b>and</b>, or <b>or</b>. <ul style="list-style-type: none"> <li>▪ The <b>not</b> operator will omit executing any test within the API Hook's scope that has been tagged with a tag listed in the <b>tagList</b>.</li> </ul> </li> </ul> </li> </ul> </li> </ul> </li> </ul>

- The **and** operator will include any test within the API Hook's scope that has been tagged with all the tags provided in the **tagList**.
- The **or** operator will include any test within the API Hook's scope that has been tagged with at least one of the tags provided in the **tagList**.

## Response Content

### MIME type: application/json

`application/json` is the default response from the endpoint.

```
[
 {
 testRunId: string,
 testId: string,
 testName: string,
 location: string,
 initiated: string,
 duration: integer,
 status: enum['passed' or 'failed'],
 reportUrl: string,
 results: {
 warningsCount: integer,
 failureCount: integer,
 httpFailures: array[object],
 criticalFailures: array[object],
 assertions: array[object]
 }
 }
]
```

### Response Description

- **testRunID**
  - ID of the test run.
- **testId**
  - ID of the test which was executed.
- **testName**
  - Name of the test.
- **location**
  - Location of the agent which executed the test.
- **initiated**
  - Date and time when the test was initiated.
- **duration**
  - Total duration of the test run in seconds.
- **status**
  - Overall result of the test.
- **reportUrl**
  - URL to the test run report.
- **results**
  - A wrapper object containing the results of the test run.
  - Child Elements:
    - **warningsCount**
      - Count of the HTTP warnings returned by the test assertions.
    - **failureCount**
      - Count of the HTTP status code responses that are greater than 400.
    - **httpFailures**
      - Description of failures whose HTTP Status response code is greater than or equal to 400.
    - **criticalFailures**
      - Description of failures whose HTTP Status response code is greater than or equal to 500.
    - **assertions**
      - A description of the assertions executed in the test. By default, only failing assertions will be listed.

### MIME type: text/xml

To receive a `text/xml` response, the optional `JUnitFormat` variable in the request body must be set to `true`. JUnit responses can be useful when you wish to incorporate your test results with a build pipeline such as Jenkins, via the [JUnit plugin](#).

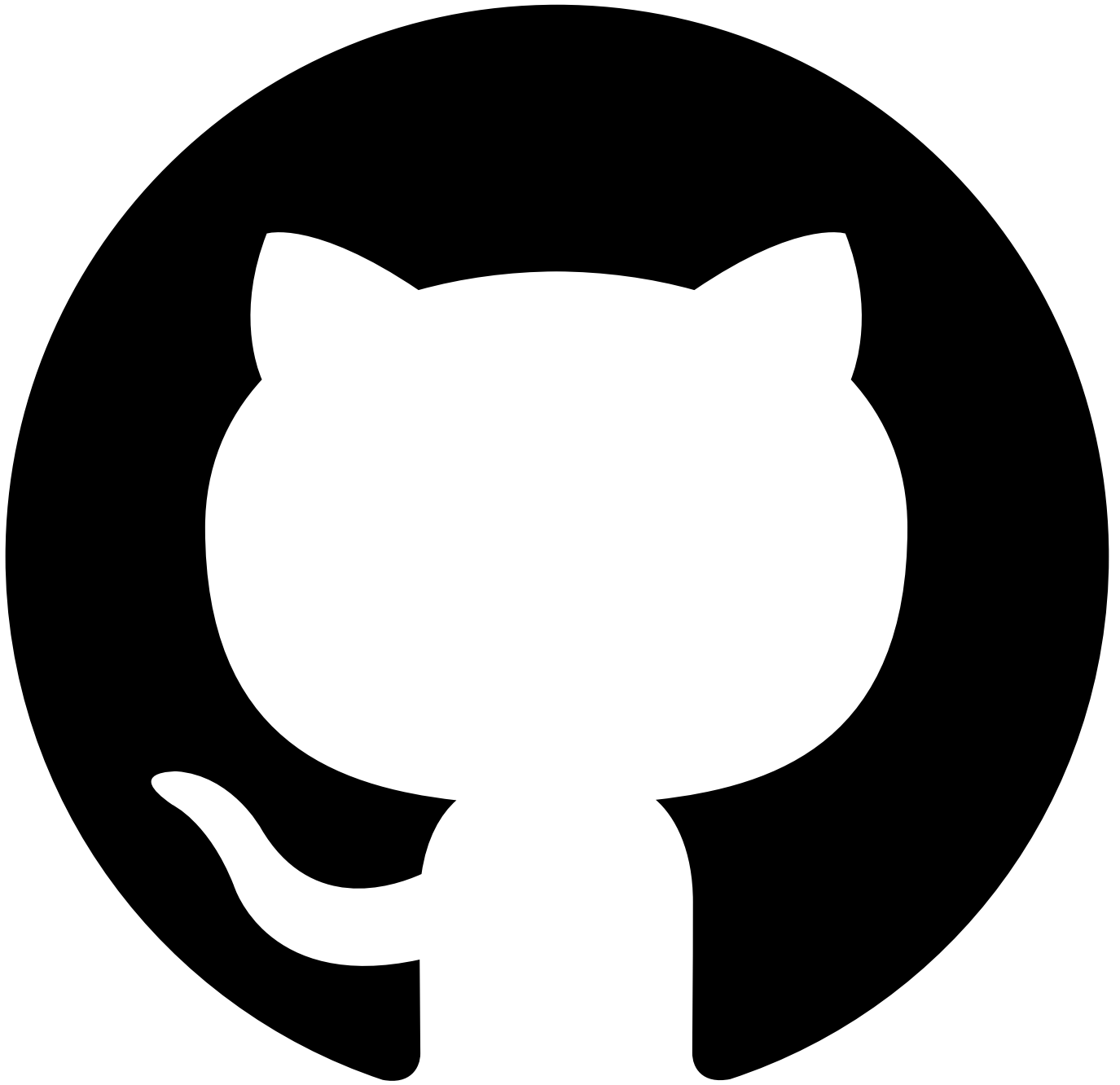
```
<?xml version="1.0" encoding="UTF-8"?>
<testSuite name="" tests="" disabled="" errors="" failures="" timestamp="" time="">
<testcase name="" classname="" time="">
 <failure message="" type="" />
</testcase>
</testSuite>
```

## Response Description

- **<testSuite>**
  - Child Elements: **<testCase>**
  - Attributes:
    - **name**
      - Test Suite name.
    - **tests**
      - Number of published tests ran.
    - **disabled**
      - Number of disabled tests.
    - **errors**
      - Number of errors.
    - **failures**
      - Number of tests within the test suite that failed.
    - **timestamp**
      - Date and time the tests were ran.
    - **time**
      - Total duration of all test runs in seconds.
- **<testCase>**
  - Child Elements: **<failure>**
  - Attributes:
    - **name**
      - Test name.
    - **classname**
      - ID of the test which was executed.
    - **time**
      - Total duration of the test run in seconds.
- **<failure>**
  - Attributes:
    - **message**
      - Failure message containing the expected and actual response.
    - **type**
      - Type of failure.

### Status Codes

200: Success  
400: Missing Authorization Headers  
401: Invalid Authorization Headers



Contribute in GitHub: [Edit online](#)

---

## Run Published Test By ID

### Description

---

The Run Published Test By ID endpoint executes a single specified test within an API Hook's scope.

### HTTP Request

---

```
POST <API-Hook-URL>/tests/{Test-ID}/run
```

---

## Sample Method Invocation

```

curl -X POST \
-H X-API-Key:{API-Key} \
-H X-API-Secret:{API-Secret} \
-H Content-Type:application/json \
-d "
 {
 "options": {
 "allAssertions": boolean,
 "JUnitFormat": boolean
 },
 "variables": {
 string: string,
 }
 }
" \
<API-Hook-URL>/tests/{Test-ID}/run

```

<b>Parameters</b>	<b>Path Parameters</b>  Test-ID: The ID of the Test to execute.  <b>Header Parameters</b>  API-Key: A valid <a href="#">API Key</a> for the Test Organization this API Hook belongs to. API-Secret: A valid <a href="#">API Secret</a> for the Test Organization this API Hook belongs to.
<b>Request Body</b>	<b>MIME type: application/json</b>  <pre> {   options: {     allAssertions: boolean,     JUnitFormat: boolean   },   variables: {     string: string,   } } </pre> <b>Body Parameters</b> <ul style="list-style-type: none"> <li>• <b>options</b> (Only required if providing a child element) <ul style="list-style-type: none"> <li>◦ A wrapper object for the containing options on how what data should be returned in the response, or how the response should be formatted.</li> <li>◦ Child Elements: <ul style="list-style-type: none"> <li>▪ <b>allAssertions</b> (Optional) <ul style="list-style-type: none"> <li>▪ When <b>false</b>, only the results of failing assertions in the test are returned. When <b>true</b>, all assertions in the test are returned. <b>allAssertions</b> is <b>false</b> by default.</li> </ul> </li> <li>▪ <b>JUnitFormat</b> (Optional) <ul style="list-style-type: none"> <li>▪ When <b>false</b>, <b>application/json</b> format is returned for the test results. When <b>true</b>, <b>text/xml</b> JUnit format is used for the test results (see <b>text/xml</b> response). <b>JUnitFormat</b> is <b>false</b> by default.</li> </ul> </li> </ul> </li> </ul> </li> <li>• <b>variables</b> (Optional) <ul style="list-style-type: none"> <li>◦ The variables section is used to pass a list of named variables and their values into the tests being run. It is only used by tests that have been written to make use of variables, and have values stored alongside them. Declaring the variable name as the object key, and its new value as the key's value (as type string), will overwrite the current value of the variable in the test which shares its name with the object key.</li> </ul> </li> </ul>
<b>Response Content</b>	<b>MIME type: application/json</b>  <b>application/json</b> is the default response from the endpoint.  <pre> [   {     testRunId: string,     testId: string,     testName: string,     location: string,     initiated: string,     duration: integer,     status: enum['passed' or 'failed'],     reportUrl: string,     results: {       warningsCount: integer,       failureCount: integer,       httpFailures: array[object],       criticalFailures: array[object],       assertions: array[object]     }   } ] </pre>



```
}
]
```

## Response Description

- **testRunID**
  - ID of the test run.
- **testId**
  - ID of the test which was executed.
- **testName**
  - Name of the test.
- **location**
  - Location of the agent which executed the test.
- **initiated**
  - Date and time when the test was initiated.
- **duration**
  - Total duration of the test run in seconds.
- **status**
  - Overall result of the test.
- **reportUrl**
  - URL to the test run report.
- **results**
  - A wrapper object containing the results of the test run.
  - Child Elements:
    - **warningsCount**
      - Count of the HTTP warnings returned by the test assertions.
    - **failureCount**
      - Count of the HTTP status code responses that are greater than 400.
    - **httpFailures**
      - Description of failures whose HTTP Status response code is greater than or equal to 400.
    - **criticalFailures**
      - Description of failures whose HTTP Status response code is greater than or equal to 500.
    - **assertions**
      - A description of the assertions executed in the test. By default, only failing assertions will be listed.

## MIME type: text/xml

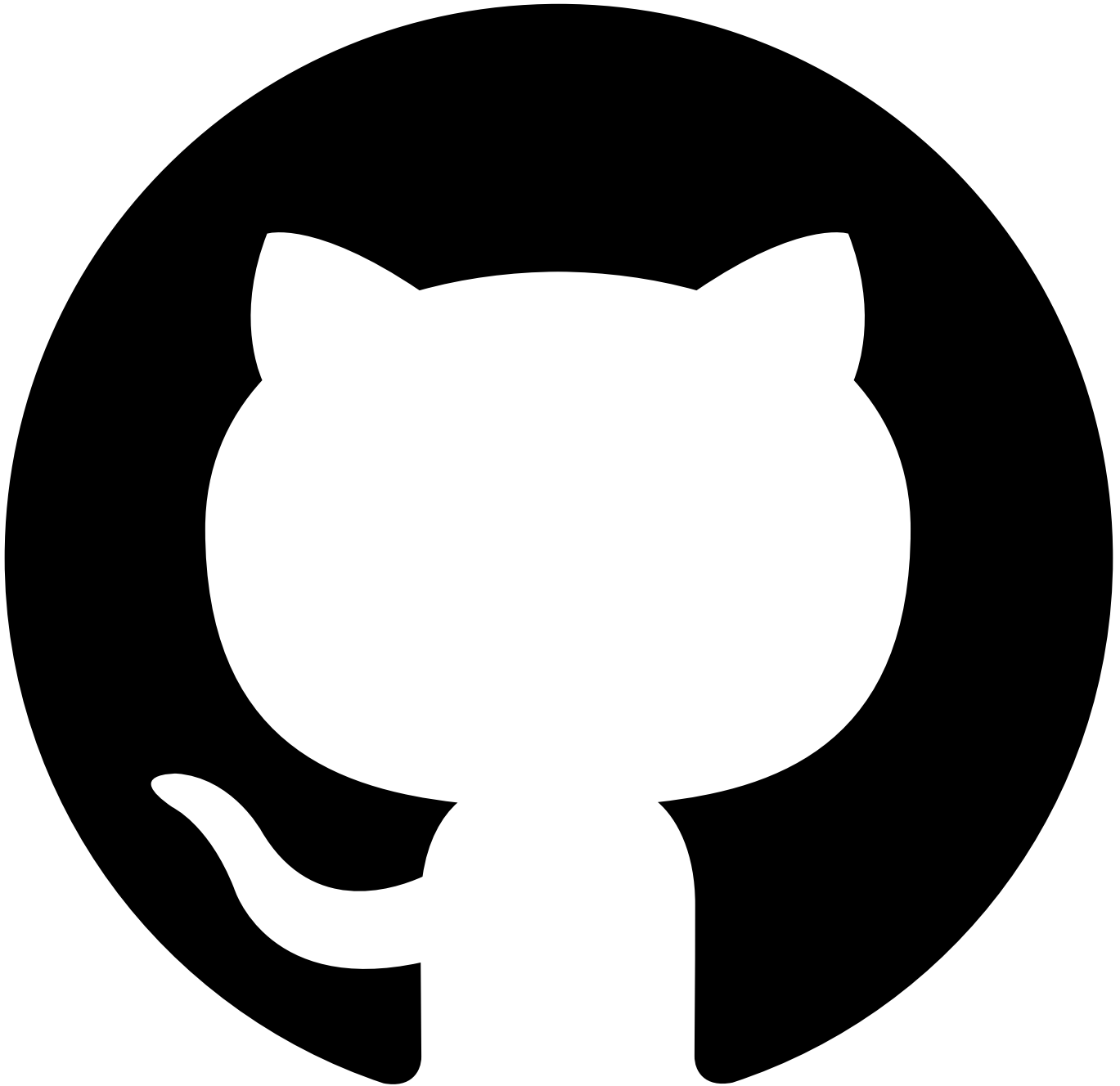
To receive a `text/xml` response, the optional `JUnitFormat` variable in the request body must be set to `true`. JUnit responses can be useful when you wish to incorporate your test results with a build pipeline such as Jenkins, via the [JUnit plugin](#).

```
<?xml version="1.0" encoding="UTF-8"?>
<testSuite name="" tests="" disabled="" errors="" failures="" timestamp="" time="">
<testcase name="" classname="" time="">
 <failure message="" type="" />
</testcase>
</testSuite>
```

## Response Description

- **<testSuite>**
  - Child Elements: **<testCase>**
  - Attributes:
    - **name**
      - Test Suite name.
    - **tests**
      - Number of published tests ran.
    - **disabled**
      - Number of disabled tests.
    - **errors**
      - Number of errors.
    - **failures**
      - Number of tests within the test suite that failed.
    - **timestamp**
      - Date and time the tests were ran.
    - **time**
      - Total duration of all test runs in seconds.
- **<testCase>**
  - Child Elements: **<failure>**
  - Attributes:
    - **name**

	<ul style="list-style-type: none"> <li>▪ Test name.</li> <li>▪ <b>classname</b> <ul style="list-style-type: none"> <li>▪ ID of the test which was executed.</li> </ul> </li> <li>▪ <b>time</b> <ul style="list-style-type: none"> <li>▪ Total duration of the test run in seconds.</li> </ul> </li> <li>• <b>&lt;failure&gt;</b> <ul style="list-style-type: none"> <li>◦ Attributes: <ul style="list-style-type: none"> <li>▪ <b>message</b> <ul style="list-style-type: none"> <li>▪ Failure message containing the expected and actual response.</li> </ul> </li> <li>▪ <b>type</b> <ul style="list-style-type: none"> <li>▪ Type of failure.</li> </ul> </li> </ul> </li> </ul> </li> </ul>
<b>Status Codes</b>	200: Success 400: Missing Authorization Headers 401: Invalid Authorization Headers



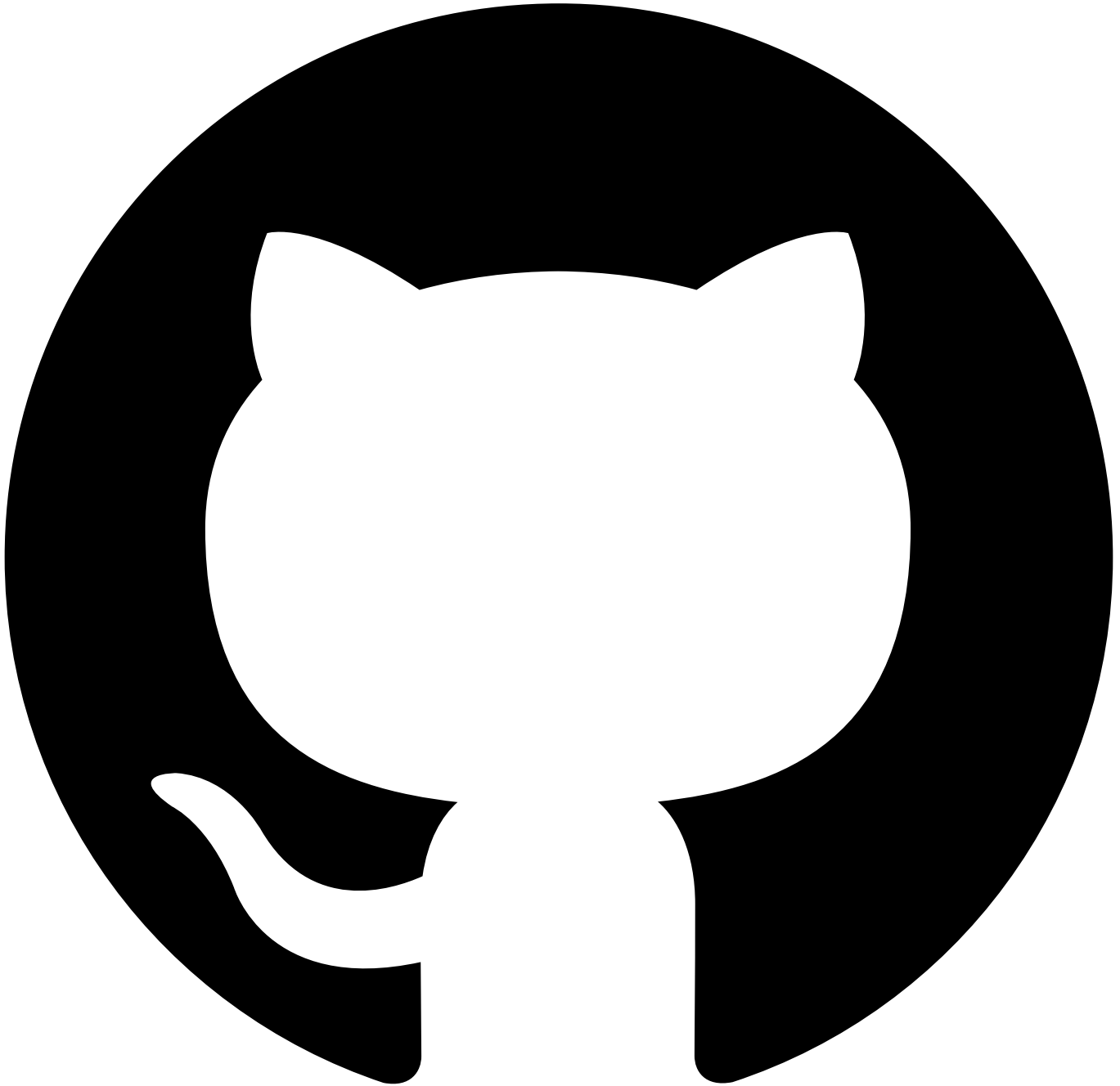
Contribute in GitHub: [Edit online](#)

---

## Automated API Behavior Testing Tutorials

Tutorials for using the Automated API behavior testing application

- [Tutorial: Invoking test cases from Jenkins with JUnit result format](#)
- [Tutorial: Linking requests using variables](#)



Contribute in GitHub: [Edit online](#)

---

## Tutorial: Invoking test cases from Jenkins with JUnit result format

### Before you begin

---

You must have the Automated API behavior testing application enabled and have created and published at least one test. You must have a Jenkins instance that you can create a new build in with the [JUnit Plugin](#) installed

### About this tutorial

---

In this tutorial, you invoke a test case using injected variables from a Jenkins pipeline

This tutorial takes you through the following steps

1. Creating an API Hook
2. Creating an API Key and Secret

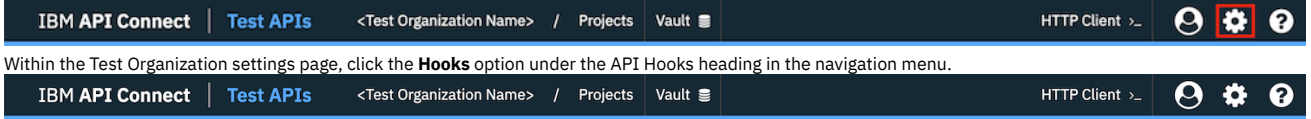
3. Creating a Jenkins build that invokes the API Hook
4. Executing the build and reviewing the test results

### Creating an API Hook

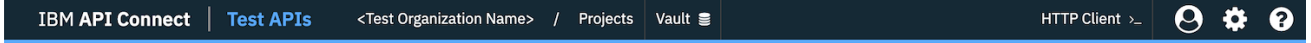
You can generate an API hook for each of the Test Suites within your Test Organization. API hooks are scoped to only one test suite and thus will only have access to operate on the data within that test suite; that is, if an API Hook is generated for **Test Suite 1**, it grants access only to the tests within **Test Suite 1**. If you wish to interact with tests in multiple test suites, you will need to generate an API hook for each Test Suite.

### How to create an API Hook

1. Navigate to the Test Organization settings page by clicking the **Cog** icon in the menu bar.

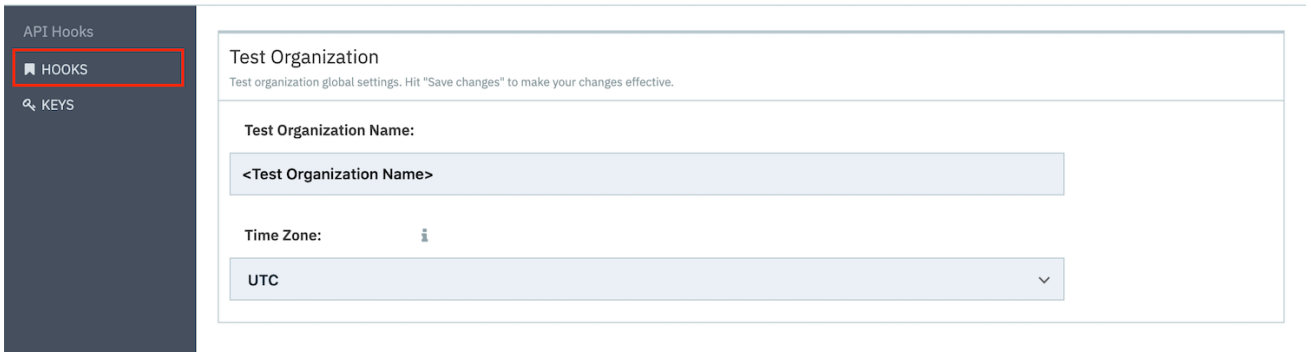


2. Within the Test Organization settings page, click the **Hooks** option under the API Hooks heading in the navigation menu.



<Test Organization Name> Settings

All changes saved.

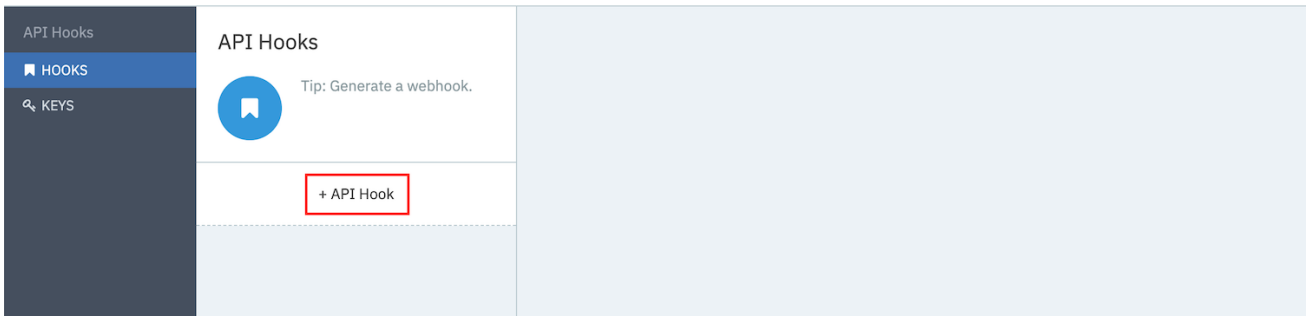



3. Now click on the **+ API Hook** button to begin creating a new API Hook.

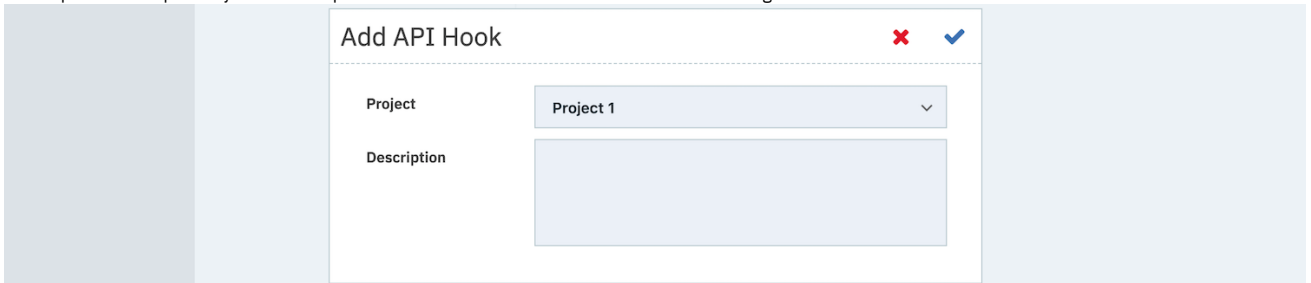


<Test Organization Name> Settings

All changes saved.



4. Because API Hooks are scoped at a Test Suite level, you will need to select the test suite you wish to generate the hook for. Choose the appropriate test suite from the dropdown and optionally add a description for the API Hook. Click the **confirm**  icon to generate and save the API Hook.



5. The API Hook URL has been created for your Test Suite. Make a note of this this URL as this will form the base path of all future API requests you make for the test suite.



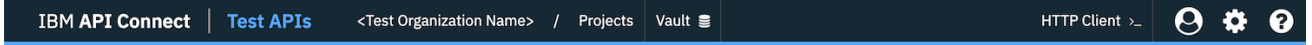
## Creating an API Key and Secret

API Keys and Secrets are the credentials required to use an [API Hook](#). For a request to an API Hook to be authorised, both the **X-API-Key** and **X-API-Secret** headers must be provided. The values of the API Key and Secret represent the values of these headers respectively.

1. First navigate to the Test Organization settings page by clicking the **Cog** icon in the menu bar.



2. Within the Test Organization settings page, click the **KEYS** option under the API Hooks heading in the navigation menu.



### <Test Organization Name> Settings

All changes saved.

API Hooks

- HOOKS
- KEYS**

#### Test Organization

Test organization global settings. Hit "Save changes" to make your changes effective.

**Test Organization Name:**

<Test Organization Name>

**Time Zone:** i

UTC

3. Now click on the **+ API Key** button to create a new API Key.



### <Test Organization Name> Settings

All changes saved.


API Hooks

- HOOKS
- KEYS**

#### API Keys

Tip: Generate an API Key.

+ API Key

4. You will then be shown the generated **Key** and **Secret**. Make a note of both of these, ensure the **Secret** is noted as it cannot be retrieved in the future. Give your new API Key a name and then click the **confirm**  icon to save it.

#### Add API Key

Name: My API Key

Key: 30eb22de-03db-40d3-8b50-d218ef449f24

Secret: ad9f94ce7b20b7a6f305df5e723b80d9c064f7f0c7cb

## Creating a Jenkins build that invokes the API Hook

1. Create a Jenkins Project or navigate to an existing one

2. Select **Configure** from the the options

**Jenkins**

Jenkins > Freestyle1 >

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now
- Delete Project
- Configure**
- Favorite
- Open Blue Ocean
- Rename

## Project Freestyle1

This is a test of CURL in Jenkins

- [Workspace](#)
- [Recent Changes](#)
- [Latest Test Result](#) (3 failures / ±0)

### Build History

trend =

find x

#30	09-Feb-2020 14:57
#29	07-Feb-2020 14:37

### Permalinks

- [Last build \(#30\), 1 day 1 hr ago](#)
- [Last stable build \(#16\), 5 days 23 hr ago](#)
- [Last successful build \(#30\), 1 day 1 hr ago](#)
- [Last failed build \(#5\), 6 days 23 hr ago](#)

3. Under **Build**, click **Add build step** dropdown and select **Execute Shell**

**Execute shell**

Command

See [the list of available environment variables](#)

Advanced...

Add build step ▾

4. Within the **Execute shell** block paste a **curl** command that will invoke the hook that was created in step 1

```
curl -XPOST
-H 'x-api-key: <API-KEY>'
-H 'x-api-secret: <API-SECRET>'
-H "Content-type: application/json"
-d '{ options: { JUnitFormat: true } }'
'<API-HOOK-URL>/tests/run'
>| testReports.xml
```

5. Under **Post-build Actions**, click **Add post-build action** and in the dropdown select **Publish JUnit test results report**.

1. For **Test report XMLs** provide the name of the report created from the curl, in this tutorial `testReports.xml`

The screenshot shows the 'Post-build Actions' configuration page in Jenkins. The 'Publish JUnit test result report' action is selected. The 'Test report XMLs' field is empty. Below it, there is a link to the 'Fileset 'includes'' setting. There are two checkboxes: 'Retain long standard output/error' (unchecked) and 'Allow empty results' (unchecked). The 'Health report amplification factor' is set to '1.0'. At the bottom, there is an 'Add post-build action' button.

6. Save the project

### Executing the build and reviewing the test results

1. Click **Build Now** on the project

The screenshot shows the Jenkins project page for 'Freestyle1'. The 'Build Now' button is highlighted with a red box. The page includes a navigation menu on the left with options like 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', 'Favorite', 'Open Blue Ocean', and 'Rename'. On the right, there is a 'Project' section with a 'Wo' folder icon and a 'Re' report icon. At the bottom, there is a 'Build History' section with a 'trend' button and a 'Permalink' section.

2. Once the build has completed, you will now see a chart showing

The screenshot shows the Jenkins interface for 'Project Freestyle1'. On the left is a navigation sidebar with options like 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', 'Favorite', 'Open Blue Ocean', and 'Rename'. The main content area includes a 'Workspace' section with links for 'Recent Changes' and 'Latest Test Result (3 failures / ±0)'. Below that is a 'Permalinks' section with a list of build links. On the right, a 'Test Result Trend' chart shows the count of successful (blue) and failed (red) tests over time. The chart shows a significant increase in failures starting around build #24, peaking at 3 failures in build #28. A 'Build History' table is also visible, showing the last three builds (#30, #29, #28) with their respective dates and times.

3. Click **Latest Test results** and review the Junit results

The screenshot shows the 'Test Result' page for build #30. At the top, it indicates '3 failures (±0)'. A progress bar shows 3 red segments (failures) and 1 blue segment (success). Below the progress bar, it says '4 tests (±0)' and 'Took 2 sec.' with an 'add description' link. The 'All Failed Tests' section contains a table with the following data:

Test Name	Duration	Age
<a href="#">5e3c151fee09b600467fd211.Greater Than</a>	0 ms	5
<a href="#">5e3c157bee09b600467fd220.Less Than</a>	0 ms	5
<a href="#">5e3940ea770acc00460b2900.Passing_Test_2</a>	0 ms	7

The 'All Tests' section contains a summary table:

Package	Duration	Fail (diff)	Skip (diff)	Pass (diff)	Total (diff)
(root)	2 sec	3	0	1	4

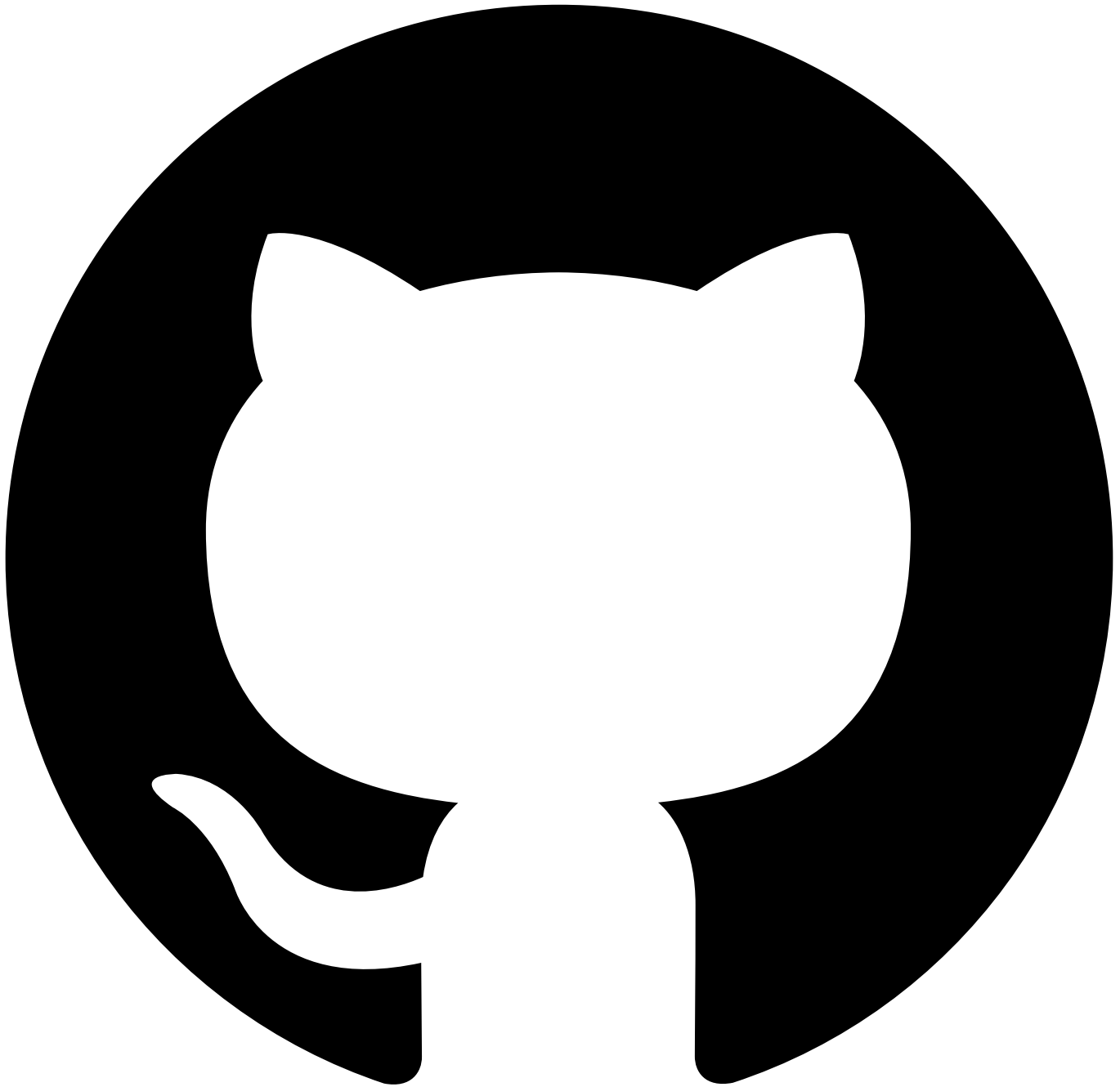
To expand upon this tutorial additional options and steps can be found under [Using API Hooks](#)

### What you did in this tutorial

In this tutorial you completed the following activities:

- Created a API Hook
- Created a Key and Secret used for authenticating with a hook
- Invoked the API Hook using the Key and Secret to start a test and process the results





Contribute in GitHub: [Edit online](#)

---

## Tutorial: Using variables within a test

### About this tutorial

---

In this tutorial, you will create a test case that uses both a variable defined in a dataset and define and use one within the test itself.

This tutorial will take you through the following testReports

1. Create a test using the HTTP Client
2. Modifying variables within data sets
3. Defining your own variable
4. Adding a new request to your test using the HTTP Client
5. Linking the requests with the variables

#### Creating a test with the HTTP Client

Using the HTTP Client it is possible to take the request and response and create a test case from these parts. The HTTP Client is accessible from either the top navigation bar or from inside a test case.

NOTE: test cases generated from the HTTP Client have no association with any specification document for the API that is being invoked, as such some details may be incorrect within the test case such as whether a given field is required or not.

### Making the request in the HTTP Client

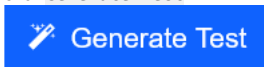
1. Go to the HTTP client using the Link
2. For the URL enter `https://<API Test hostname>/apitest/api/exmaples/retail/products`, this is an unauthenticated **GET** request, as such no additional values are required
3. Click **Send**
4. Once the response has returned you should see this:

The screenshot shows the HTTP Client interface. At the top, there are buttons for 'Save', 'Generate Test', and 'Import'. Below that, the request method is set to 'GET' and the URL is 'https://hub.atm-test.ibm.com/apitest/api/examples/retail/products'. There are 'Params', 'Send', and 'Clear' buttons. The 'Response' section shows a JSON object: 

```
{ "success": true, "status": 200, "content": { "act": "list", "products": [{ "id": 111, "name": "Product 1" }, { "id": 222, "name": "Product 2" }] } }
```

. The status bar indicates 'HTTP Code: 200', 'Latency: 35', 'Fetch: 188', and 'Total Time: 223'.

5. Click **Generate Test**



6. Provide a Test name and either add a new Test Suite or select an existing one

The 'Generate Test' dialog box has a title bar with a red 'X' and a blue checkmark. It contains two main sections. The first section has a 'Name' label and a text input field containing 'Retail Products'. The second section has a 'Save to Test Suite' label and a dropdown menu with 'Create new test suite' selected. Below the dropdown is a list of existing test suites: 'Retail Examples'.

7. Click save



This will automatically generate a test case based on your request and you should have a **GET** request with a number of assertions

### Modifying variables within a Data sets

Having generated the test, you will see that the request will have a URL target of `${endpointURL}` this is defined within the Global variables under Data Sets

In cases of tests generated tests, the value of `endpointURL` is the complete URL that we called, however if we want to call other paths on that endpoint we will need to modify it so we can reuse it in cases where call a different URL

This is what we will have from the generated test from the previous step:


The endpoint currently uses the full path of `https://<API Test hostname>/apitest/api/exmaples/retail/products`, but later I will make a call to a different resource to `products`, so I want to trim this to make it more reusable

1. Selecting Edit variable on `endpointURL`

2. Update the value to be `https://<API Test hostname>/apitest/api/exmaples/retail`
3. Click **Save**

As we have updated the value of the input, we now need to fix the request as it will be missing `/products` from the endpoint

1. Click **Edit** on the **GET** Request
2. Add `/products` after the `${endpointURL}`
3. Click save

Have made the updates we can check the test is still valid by clicking  image of Run Test

and we should get the payload of

```
{
 "success": true,
 "status": 200,
 "content": {
 "act": "list",
 "products": [
 511,
 611,
 711
]
 }
}
```

## Defining your own variable

The next step is to dig into the details of an individual product, based on one of the products ID we get back from `/products`

The IDs are process within the test case within the `For each loop` and accessible as `_1`, if we wanted to grab the `status` value we would to it in the way that assertion refers to it `payload.status`

1. Click **Add Components**



2. Click **Set (variable)**



**Set (variable)**

3. Provide the name of variables such as `productID`
4. Due to the processing the above values are treated as `floats`, but for the next call we need an integer a. Select **Language** for Mode b. Within **Content** provide the following groovy script `return Math.round(_1)` - this converts the float to an into
5. Move Set Variable to within the For Each loop

## Adding a New request to an existing test with the HTTP client

Once you are in the Test Editor there are two ways to add a new request, either adding one from the list of components or by making or loading a request within the client within the test Editor

for this tutorial we will perform the request in client inside the test


1. Click HTTP Client icon



**HTTP Client**

2. In the URL field enter `https://<API Test hostname>/apitest/api/exmaples/retail/product?id=711`
3. Click **Send**
4. Click **Generate Test**



5. Click **Continue** on each screen and finally **close** - generation of endpointURL variable will fail as it already exists, this is to be expected
6. Edit the URL value in the new request to be `${endpointURL}/product`
7. Edit the Query Parameters **ID** reference variable to be the one we created in the previous stage `productID`
8. Click 



You have now completed a linked set of requests, to check that this is running correctly you can now run the test and check the request to see what parameter value was used



GET  
get https://hub.atm-test-master.fyre.ibm.com/apitest/api/examples/retail/product?id=711

Latency: 25ms      Download: 24ms

STATUS: 200

[See Request](#)

## Testing an API with the Policies editor

IBM® API Connect provides a basic test environment in the Policies editor so that you can ensure that your APIs are defined and implemented correctly.

## About this task




You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

Note:

- If you are testing an OpenAPI 2.0 API, you can use either the Policies editor test tool described here, or you can use the Test tab. If you are testing an OpenAPI 3.0 API, only the Test tab is available; see [Using the Test tab to debug your API](#).
- If you are testing an API that contains references to API properties, only those references that are defined inside the API assembly are resolved and replaced with their corresponding values when you invoke the API in the assembly test tool; property references that are defined outside of the API assembly are not resolved. For more information on API properties, see [Setting API properties](#) (OpenAPI 2.0) or [Setting API properties](#) (OpenAPI 3.0).
- Due to Cross-Origin Resource Sharing (CORS) restrictions, the assembly test tool cannot be used with the Chrome or Safari browsers on the macOS Catalina platform.

## Procedure

To test an API, complete the following steps.

1. If you are using API Designer, set the mode to Online using the Options menu  on the main page.
2. In the navigation pane, click  Develop, then select the APIs tab.
3. Click the title of the API that you want to work with.
4. Click the Gateway tab then, in the navigation pane, click Policies.
5. If you are testing the API for the first time and, when you created the API definition, you selected the Activate API option, your test setup will already be configured and you can proceed immediately to the next step to test your API. Otherwise, click Offline to toggle the API to the Online state. If you are retesting your API after making changes, the modified API will have been republished automatically on saving.
6. Click the Test icon .
 

Note:

  - The Test icon is active only if you have toggled your API to the Online state, as described in step 5.
  - The Catalog, Product, Plan, and application details displayed in the Setup pane reflect your configured API testing preferences; see [Specifying the testing preferences for an API](#).
7. In the Operation section, select the API operation that you want to test, then click Invoke.  
The API response is displayed in the Response section.  
Note: If you receive a message relating to an untrusted certificate, click the link provided, accept the certificate, then return to the test environment and click Invoke again. The message also mentions a lack of CORS support on the server, but this is just one possible cause for the connection failing.

## Testing an API with the Local Test Environment

Use the Local Test Environment to test APIs on your local machine, without the need to connect to an API Connect management server. The Local Test Environment is a lightweight API Manager running on your local machine. It allows you to rapidly test APIs locally.

API Connect provides the following methods for testing an API on your local machine:

- Invoke the API from the API Designer UI application running in Online mode as described in [Testing an API](#).
- Call the API in the Local Test Environment with a cURL command, as described in the following sections.

## Prerequisites

- The API Connect developer toolkit, including the API Designer user interface, installed. For installation and running instructions, see [Installing the toolkit](#).
- The Local Test Environment and the API Designer must be from the same API Connect fix pack release in order for them to work together.
- Docker installed.  
Note: The Local Test Environment is not supported with Docker Version 18.09.x.
- A minimum of 4 GB of RAM available to Docker if a single gateway type is used, or 6 GB if both the DataPower® API Gateway and DataPower Gateway (v5 compatible) are used.  
Note: As you increase the number of APIs that are published to your gateways, you will need to allocate further memory to Docker. You will also need to start the Local Test Environment with a larger database; see [apic-lte start](#).
- If you are using Windows, ensure that your C: drive (or the drive on which your HOME directory is located, if different), is enabled as a shared drive so that the Local Test Environment files are accessible to the Docker containers.

## Installing the Local Test Environment

There are two options for installing the Local Test Environment:

- Each user downloads the Local Test Environment images to their local machine and installs the Local Test Environment from there.
- One user downloads the Local Test Environment images and uploads them to a private Docker registry, from where any user can install the Local Test Environment.

To install the Local Test Environment from your local machine, complete the following steps:

1. Open a browser and visit the [API Connect announcement page](#), search for your product version, and locate the "Downloads" table on the product announcement page. In the table, click the link to download each of the following files:
  - `apic-lte-images-version.tar.gz`, which contains all required Docker images.  
Example: `apic-lte-images-10.0.2.tar.gz`
  - `apic-lte-platform-version`, which are binary files for the Mac OS X, Linux®, and Windows platforms.  
Examples:
    - Mac OS X: `apic-lte-osx-10.0.2`
    - Linux: `apic-lte-linux-10.0.2`
    - Windows: `apic-lte-win-10.0.2`
2. On the Mac OS X or Linux platform, use the `chmod` command to make the binary file executable; for example:

```
chmod +x linux-apic-lte
```

3. Load the Docker images into your local Docker image repository by entering the following command:

```
docker load < apic-lte-images.tar.gz
```

Note: In all Local Test Environment commands that are used on the remainder of this page, replace *platform* with **osx**, **linux**, or **windows**, depending on your platform, as follows:

- Mac OS X: replace **platform-apic-lte** with **osx-apic-lte**
- Linux: replace **platform-apic-lte** with **linux-apic-lte**
- Windows: replace **platform-apic-lte** with **win-apic-lte**

To upload the Local Test Environment images to a private Docker registry, complete the following steps:

1. Download the **IBM\_API\_CONNECT\_LOCAL\_TEST\_ENVIRO.zip** file as described in step 1.
2. Distribute the appropriate binary file to all users, according to the platform.
3. Upload the Local Test Environment to your private Docker registry; enter the following command:

```
platform-apic-lte registry-upload apic-lte-images.tar.gz registry_host
```

where *registry\_host* is the host name or IP address of your private Docker registry. Now, any user can install and run the Local Test Environment as follows:

- a. If the private Docker registry requires authentication, log in by entering the following command:

```
docker login registry_host
```

- b. Load the Docker images into your local Docker image repository by entering the following command:

```
platform-apic-lte init registry_host
```

## Starting the Local Test Environment

1. Start the Docker images by entering the following command:

```
platform-apic-lte start
```

Note:

- By default, the **platform-apic-lte start** command starts only a DataPower API Gateway. To also start a DataPower Gateway (v5 compatible), enter the following command:

```
platform-apic-lte start --datapower-gateway-enabled --datapower-api-gateway-enabled
```

- The Local Test Environment might fail to start with an error message that includes the strings **Error: certificate is not yet valid** and **CERT\_NOT\_YET\_VALID**. The most likely cause is that the date and time setting is incorrect on the machine that is running the Local Test Environment. Ensure that the date and time setting is correct, before attempting the start command again. If you are using Docker for Windows, the clock in the Docker containers can become out of sync with the system clock, especially after a machine has been put in sleep mode. In this case, restarting Docker should fix the clock discrepancy; for more information, see <https://github.com/docker-for-win/issues/4526>.
- By default, the LTE starts with an empty backend database that does not contain the APIs and Products that might have been published during an earlier run of the LTE. To start the LTE with the backend database that was used during the previous run, use the flag **--keep-config**, for example, **platform-apic-lte start --keep-config**. When you use **--keep-config** any other flag that is specified for the start is ignored. Instead the same flags that were used during the earlier start are used, in particular the same gateways are enabled.

2. Verify that the Local Test Environment is installed and running correctly by entering the following commands:

- a. **platform-apic-lte status**

This output from this command shows the status of all components, and provides endpoint and authentication details, and should be similar to the following:

```
Container Status

apic-lte-apim Up 3 minutes
apic-lte-datapower-gateway Not Running
apic-lte-datapower-api-gateway Up 2 minutes
apic-lte-db Up 3 minutes
apic-lte-juhu Up 3 minutes
apic-lte-lur Up 3 minutes

- Platform API url: https://localhost:2000
- Admin user: username=admin, password=7iron-hide
- 'localtest' org owner: username=shavon, password=7iron-hide
- 'localtest' org sandbox test app credentials client id: 80963e74076afe50d346d76401c3c08a
- Datapower API Gateway API base url: https://localhost:9444/localtest/sandbox/
```

- b. **apic login --server localhost:2000 --username shavon --password 7iron-hide --realm provider/default-idp-2**

This command confirms that you can log in to the management server, and the response should be as follows:

```
Logged into localhost:2000 successfully
```

Note: If you have previously installed toolkit credentials, as detailed in [Installing the toolkit](#), attempting to log in to the Local Test Environment management server from the toolkit CLI will fail; you must first clear the credentials by using the following command:

```
apic client-creds:clear
```

The toolkit CLI will now use the default credentials for login to the toolkit Local Test Environment. If you subsequently log in to a management server in an API Connect cloud, the default toolkit credentials will also be used, rather than the credentials that are unique to that deployment. To use the unique credentials, re-install them by using the following command:

```
apic client-creds:set toolkit_credentials_file_path/credentials.json
```

where *credentials\_file\_path* is the location of the previously downloaded toolkit credentials JSON file. To determine whether you currently have any toolkit credentials installed, use the following command:

## Preparing an API for testing in the Local Test Environment

To prepare an API for testing in the Local Test Environment, you must publish it to the Sandbox Catalog in the Local Test Environment. If you want to test an API that you already published, proceed to [Testing an API in the Local Test Environment](#), otherwise, complete the following steps:

1. Launch the API Designer user interface.
2. Open the required local directory; this is the directory in which your API and Product definition files will be stored.
3. Connect to the Local Test Environment. If you haven't previously connected to the Local Test Environment, click Add Another Cloud, then complete the following steps:
  - a. In the HOST URL field, enter `https://localhost:2000`, then click Next.
  - b. In the Username field, enter `shavon`, in the Password field enter `7iron-hide`, then click Sign in.
 If you have previously connected to the Local Test Environment, click the existing tile to log in immediately.

The API Designer welcome page opens.

4. Click Develop APIs and Products, then click the API that you want to test. For details on how to configure an API definition, see [Developing your APIs and applications](#).
5. Move the activation slider control to the on position:



When the publish operation completes, your API is ready for testing.

Note: Whenever you make any changes to an API, you must republish it before retesting.

## Testing an API in the Local Test Environment.

To test an API in the Local Test Environment, issue a REST API call to the following URL:

`https://localhost:9444/localtest/sandbox/basepath/operation_path?client_id=lte_client_id`

where:

- `basepath` is the base path that is configured in the API definition.
- `operation_path` is the path for the operation that you want to invoke, as configured in the API definition.
- `lte_client_id` is the client ID for the test application in the local test environment, as returned by the `platform-apic-lte status` command in step 2.

The following example show how to test the API that is created in the tutorial [Creating a proxy REST API definition](#), by using the `curl` utility; the API returns the details of bank branches:

```
curl -k https://localhost:9444/localtest/sandbox/branches/details?client_id=80963e74076afe50d346d76401c3c08a
[{"id": "0b3a8cf0-7e78-11e5-8059-a1020f32cce5", "type": "atm", "address": {"street1": "600 Anton Blvd.", "street2": "Floor 5", "city": "Costa Mesa", "state": "CA", "zip_code": "92626"}}, {"id": "9d72ecec-7e7b-11e5-9038-55f9f9c08c06", "type": "atm", "address": {"street1": "4660 La Jolla Village Drive", "street2": "Suite 300", "city": "San Diego", "state": "CA", "zip_code": "92122"}}, {"id": "ae648760-7e77-11e5-8059-a1020f32cce5", "type": "atm", "address": {"street1": "New Orchard Road", "city": "Armonk", "state": "NY", "zip_code": "10504"}}, {"id": "c23397f0-7e76-11e5-8059-a1020f32cce5", "type": "branch", "phone": "512-286-5000", "address": {"street1": "11400 Burnet Rd.", "city": "Austin", "state": "TX", "zip_code": "78758-3415"}}, {"id": "ca841550-7e77-11e5-8059-a1020f32cce5", "type": "atm", "address": {"street1": "334 Route 9W", "city": "Palisades", "state": "NY", "zip_code": "10964"}}, {"id": "dcl32eb0-7e7b-11e5-9038-55f9f9c08c06", "type": "branch", "phone": "978-899-3444", "address": {"street1": "550 King St.", "city": "Littleton", "state": "MA", "zip_code": "01460-1250"}}, {"id": "e1161670-7e76-11e5-8059-a1020f32cce5", "type": "branch", "phone": "561-893-7700", "address": {"street1": "5901 Broken Sound Pkwy. NW", "city": "Boca Raton", "state": "FL", "zip_code": "33487-2773"}}, {"id": "f9ca9ab0-7e7b-11e5-9038-55f9f9c08c06", "type": "atm", "address": {"street1": "1 Rogers Street", "city": "Cambridge", "state": "MA", "zip_code": "02142"}}]
```

## Local Test Environment commands

The following table summarizes the Local Test Environment commands; use the `help` command to get full usage details for any command.

Table 1. Local Test Environment command summary

Command	Description
<code>platform-apic-lte help command</code>	Display help information for any command.
<code>platform-apic-lte init</code>	Download the Local Test Environment Docker images.
<code>platform-apic-lte start</code>	Start the Local Test Environment Docker images. Use the <code>--database-max-heap-size</code> parameter to set the size of the Local Test Environment database, in bytes; for example:  <pre>linux-apic-lte start --database-max-heap-size 4096M linux-apic-lte start --database-max-heap-size 1G linux-apic-lte start --database-max-heap-size 1048576K linux-apic-lte start --database-max-heap-size 1073741824</pre> The default value is 1024M. Tip: By default, the <code>platform-apic-lte start</code> command deletes all previous data and re-initializes the Local Test Environment configuration, so all your previous configuration, including published Products, is deleted. To retain the previous configuration, and to apply the same command parameters that were used in the previous <code>platform-apic-lte start</code> command, supply the <code>--keep-config</code> parameter.
<code>platform-apic-lte status</code>	Display status information for the Local Test Environment components, and endpoint and authentication details.

Command	Description
<code>platform-apic-lte stop</code>	Stop the Local Test Environment Docker images.
<code>platform-apic-lte version</code>	Display Local Test Environment version information.

## Troubleshooting the Local Test Environment

You can consult the log file for each Local Test Environment microservice or database by using the following command:

```
docker logs container-name
```

where *container-name* is one of the following:

- `apic-lte-juhu`: the authentication gateway
- `apic-lte-apim`: the API Management service
- `apic-lte-lur`: the Local User Registry
- `apic-lte-db`: the Postgres database of the API Management service
- `apic-lte-datapower-api-gateway`: the DataPower API Gateway
- `apic-lte-datapower-gateway`: the DataPower Gateway (v5 compatible)

You can access the gateway logs in either of the following ways:

- Use the gateway administration web UI:
  1. Open the page `https://localhost:web_ui_port` in a browser; for details of the required port value, see [Local Test Environment port values](#).
  2. Select the apiconnect domain and the WebGUI interface, and log in with user name `admin` and password `admin`.
  3. Click View Logs.
- Use the gateway administration CLI:
  1. Open an SSH connection by using the following command:
 

```
ssh -p gateway-ssh-port localhost
```

For details of the required port value, see [Local Test Environment port values](#). The user name is `admin` and the password is `admin`.
  2. Enter the command `switch domain apiconnect`.
  3. To view the gateway log, enter the command `show log`.
  4. To view the log for the communication between the gateway and the API management system, enter the command `show logging gwd-log`.

## Local Test Environment port values

If any of the default port values for the Local Test Environment components conflict with ports already in use on your system, you can change them when you start the Local Test Environment by passing one or more `--component port_value` parameters to the `platform-apic-lte start` command, where:

- *component* is the Local Test Environment component whose port value you want to change.
- *port\_value* is the required value.

For example:

```
platform-apic-lte start --datapower-api-gateway-api-port 9445
```

The following table lists the components, together with the corresponding *component* parameters, and the default port values:

Component	component parameter	Default port value
DataPower API Gateway API port	<code>datapower-api-gateway-api-port</code>	9444
DataPower API Gateway API Connect service port	<code>datapower-api-gateway-apic-service-port</code>	3001
DataPower API Gateway REST management port	<code>datapower-api-gateway-rest-management-port</code>	5555
DataPower API Gateway SSH port	<code>datapower-api-gateway-ssh-port</code>	9023
DataPower API Gateway administration web UI	<code>datapower-api-gateway-web-gui-port</code>	9091
DataPower API Gateway XML management port	<code>datapower-api-gateway-xml-management-port</code>	5551
DataPower Gateway (v5 compatible) API port	<code>datapower-gateway-api-port</code>	9443
DataPower Gateway (v5 compatible) API Connect service port	<code>datapower-gateway-apic-service-port</code>	3000
DataPower Gateway (v5 compatible) REST management port	<code>datapower-gateway-rest-management-port</code>	5554
DataPower Gateway (v5 compatible) SSH port	<code>datapower-gateway-ssh-port</code>	9022
DataPower Gateway (v5 compatible) administration web UI	<code>datapower-gateway-web-gui-port</code>	9090
DataPower Gateway (v5 compatible) XML management port	<code>datapower-gateway-xml-management-port</code>	5550
Platform API port	<code>platform-api-port</code>	2000

## Related information

- [Creating TLS Client Profile in the Local Test Environment](#)

## Testing an API with the Explorer tab

Use the Explorer tab in the API editor to see how your APIs look to a consumer in the Developer Portal. You can also use the Try it tool to test the behavior of the API.



## Before you begin

To test an API in the Try it test tool, the API must be auto-published. If the API is not auto-published, the Try it tool isn't enabled in the Explorer.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

## About this task



The Explorer tab in the API editor of the API Designer UI and API Manager UI enables you to view how your APIs look to an API consumer in the Developer Portal. You can check the descriptions of the different artifacts, and as well as any schemas or examples, and you can also test the behavior of the API by using the Try it tool.

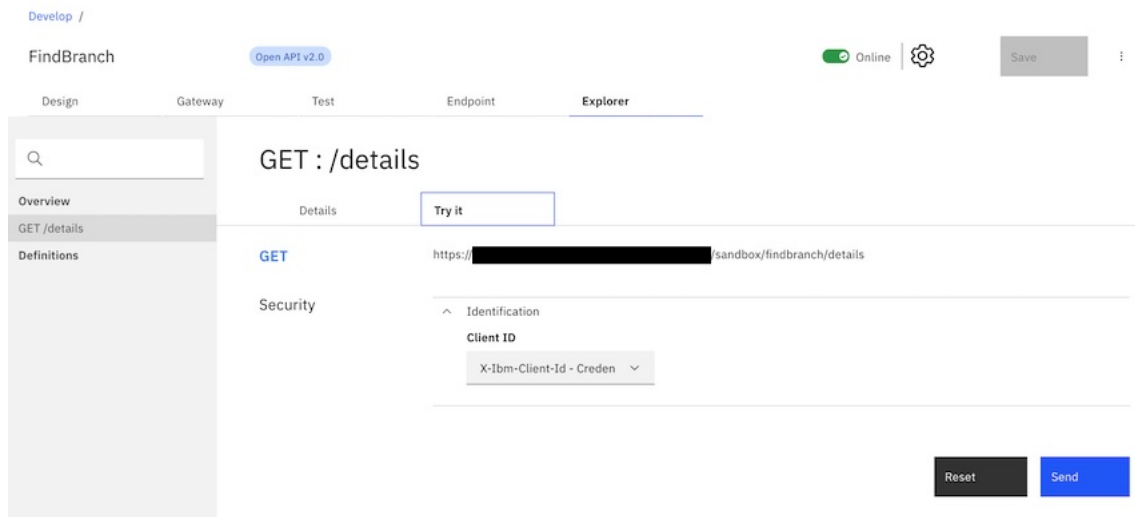
Note:

- You cannot use the Try it tool to test non-enforced APIs.

## Procedure

To test an API by using the Try it tool, complete the following steps.

- If you are using API Designer, set the mode to Online using the Options menu  on the main page.
- In the navigation pane, click  Develop, then select the APIs tab.
- Click the title of the API that you want to work with.  
The API editor is displayed.
- Click the Explorer tab.  
An Overview of the API is displayed.
- Click the API artifacts in the side menu bar to explore the operations and definitions that are available in the API, including reviewing any examples and schemas.
- To test API behavior, click an operation in the side menu bar, and click the Try it tab.  
The test pane for the operation is displayed, as shown in the following example:



- Provide any required API parameters, and click Send.  
The API response is displayed in the Response section.  
Note: If you receive a message relating to an untrusted certificate, click the link provided, accept the certificate, then return to the test environment and click Send again.

## What to do next

When you are happy with the look and behavior of your API, you can stage the API to a Product in a Catalog. For more information, see [Staging an API](#).

## Staging an API

The graphical wizard provides an option that adds the API to a Product and stages the Product in a Catalog. When a Product is in the staged state, it is not yet visible to, or subscribable by, any developers. The syndication feature in IBM® API Connect means that if Spaces are enabled for a Catalog, Products can be staged only to a Space within that Catalog.

## Before you begin

Ensure that you have a Catalog to stage to in the API Manager or API Designer user interfaces (UI). For more information, see [Creating and configuring Catalogs](#).

Ensure that the Catalog has at least one gateway service configured.

Note: All references in this topic to a Catalog can also be applied to Spaces in a Catalog, unless specified otherwise. For more information about Spaces, see [Using syndication in IBM API Connect®](#).

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Product > Stage permission for the target Catalog or Space. For information on configuring Product management permissions for a Catalog or Space, see [Creating and configuring Catalogs](#) or [Managing user access in a Space](#).

## About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI. Staging is not available when working offline in API Designer.

A Catalog is a staging target, and behaves as a logical partition of the DataPower® Gateway, and the Developer Portal.



Validation of the API definition file occurs during the staging or publishing process. The following validation occurs:

- Validation against the OpenAPI schema by using the API Dev Tools Swagger Parser (<https://www.npmjs.com/package/@apidevtools/swagger-parser>).
- Validation against IBM extension properties.
- Semantic validation, which includes the following types of validation:
  - Ensuring that if an OpenAPI is enforced by an API Connect Gateway, then the scheme must be HTTPS, or the parameter name for an API key security scheme in the header must be either **X-IBM-Client-Id** or **X-IBM-Client-Secret**.
  - Ensuring that if the OpenAPI is not enforced by an API Connect Gateway, then a "host" must be provided.
  - De-referencing of local references in the definition file (that is, values of **\$ref** properties), and ensuring these are valid JSON Pointers within the file.

Note: If you are using Event Endpoint Management in Cloud Pak for Integration, similar validation is carried out for AsyncAPI documents. For more information, see the section about API document and enforcement in the [Cloud Pak for Integration documentation](#).

Note: If the OpenAPI file that defines your API uses a **\$ref** field to reference a fragment of OpenAPI code that is defined in a separate file, the **\$ref** field is replaced with the contents of the target file before the product that contains the API is staged or published (the **\$ref** field is supported only if you are using the API Connect for IBM Cloud developer toolkit). For more information, see [Using \\$ref to reuse code fragments in your OpenAPI files](#).


## Procedure

1. In the navigation pane, click  Develop, then select the APIs tab.  
The Develop page opens.
2. **Optional:** If you have accounts on multiple provider organizations, you can select a new provider organization for staging and publishing from the Organization menu.
3. You can stage an API either from the Develop listing page, or from within the API definition itself.
  - a. To stage an API from the Develop listing page, click the options menu icon  alongside the required API, and then select Stage.
  - b. To stage an API from within the API definition, complete the following steps:
    - i. Click the title of the API that you want to work with.
    - ii. Click the options menu icon:



- iii. Click Stage.
4. Choose one of the following actions:
    - To stage the API by adding it to a new Product:
      - Select New Product - Publish using a new product.
      - When prompted, enter a Title and Version.
      - The Product Name is automatically entered. The Name is the used to refer to the product in CLI commands. See [apic products](#).
      - Click Next.
    - To stage the API by adding it to an existing Product:
      - Select Existing Product - Publish using an existing product
      - Select the Product you want to use.
      - Click Next.
  5. On the Stage To page, select the Catalog to which you want to stage the Product.  
Note: The Catalogs that you can select from are those that are defined for the management server and provider organization that you are connected to. If you are using the API Manager user interface, the connection details are determined by the API Manager URL that you open, and the user ID with which you log in. If you are using the API Designer user interface, you provide the management server details and user ID in the login window that opens when you first launch API Designer; see [Logging into API Connect Designer](#).  
  
For details of how to create a Catalog in a provider organization, see [Creating and configuring Catalogs](#).
  6. If, when the staged Product is subsequently published, you want it to be published only to selected gateway services, select Publish to specific gateway services, then select the required gateway services. Only the gateway services whose type matches the gateway type setting for the Product are listed. For information on gateway types, see [API Connect gateway types](#).
  7. Click Stage.

## Results

Your Product is staged to a Catalog. You can view the state of the Product in the Catalog in API Manager. If you staged the product from API Designer, ensure you are logged into API Manager with the same user name and password that you used for API Designer. Click  Manage in the API Manager UI, then select the required Catalog. The Product is shown with a state of Staged.

For information about the lifecycle of a product, see [The Product lifecycle](#).

If approval is required to stage Products in the Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is staged when the request is approved. If approval is not required, the Product is staged immediately.

For information on configuring Product lifecycle approvals for a Catalog, see [Creating and configuring Catalogs](#). For information on approving requests, see [Approving Product lifecycle and subscription requests](#).

If the Product contains no Plans, a Plan called Default Plan is added automatically to the Product in the Catalog.

---

## Publishing an API

The user interface provides an option that adds an API to a Product and publishes the Product in a Catalog. Publishing a draft Product makes the APIs in that Product visible on the Developer Portal for use by application developers. The syndication feature in IBM® API Connect means that if Spaces are enabled for a Catalog, Products can be published only to a Space within that Catalog.

---

### Before you begin

Ensure that you have a Catalog to stage to in the API Manager or API Designer user interfaces (UI). For more information, see [Creating and configuring Catalogs](#).

Ensure that the Catalog has at least one gateway service configured.

Note: All references in this topic to a Catalog can also be applied to Spaces in a Catalog, unless specified otherwise. For more information about Spaces, see [Using syndication in IBM API Connect®](#).

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Product Management permission for the target Catalog or Space. For information on configuring Product management permissions for a Catalog or Space, see [Creating and configuring Catalogs](#) or [Managing user access in a Space](#).

---

### About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI. Publishing is not available when working offline in API Designer.

Validation of the API definition file occurs during the staging or publishing process. The following validation occurs:

- Validation against the OpenAPI schema by using the API Dev Tools Swagger Parser (<https://www.npmjs.com/package/@apidetools/swagger-parser>).
- Validation against IBM extension properties.
- Semantic validation, which includes the following types of validation:
  - Ensuring that if an OpenAPI is enforced by an API Connect Gateway, then the scheme must be HTTPS, or the parameter name for an API key security scheme in the header must be either `X-IBM-Client-Id` or `X-IBM-Client-Secret`.
  - Ensuring that if the OpenAPI is not enforced by an API Connect Gateway, then a "host" must be provided.
  - De-referencing of local references in the definition file (that is, values of `$ref` properties), and ensuring these are valid JSON Pointers within the file.



Note: Products that contain an API with a Swagger property using `regex` that include lookahead assertions, such as "(?)" cannot be validated or published. An error message is returned. For example:

```
Product has not been published!
The multipart 'openapi' field contains an OpenAPI definition with validation errors.
definitions.properties.pattern Does not match format 'regex' (context: (root).definitions.properties.pattern, line: 0, col:
0)
400
```

Note: If the OpenAPI file that defines your API uses a `$ref` field to reference a fragment of OpenAPI code that is defined in a separate file, the `$ref` field is replaced with the contents of the target file before the product that contains the API is staged or published (the `$ref` field is supported only if you are using the API Connect for IBM Cloud developer toolkit). For more information, see [Using \\$ref to reuse code fragments in your OpenAPI files](#).

---

## Procedure

1. In the navigation pane, click  Develop, then select the APIs tab.  
The Develop page opens.
2. **Optional:** If you have accounts on multiple provider organizations, you can select a new provider organization for staging and publishing from the Organization menu.
3. You can publish an API either from the Develop listing page, or from within the API definition itself.
  - a. To publish an API from the Develop listing page, click the options menu icon  alongside the required API, and then select Publish.
  - b. To publish an API from within the API definition, complete the following steps:
    - i. Click the title of the API that you want to work with.
    - ii. Click the options menu icon:



iii. Click Publish.

4. Choose one of the following actions:


- To publish the API by adding it to a new Product:
  - Select New Product - Publish using a new product.
  - When prompted, enter a Title and Version.
  - The Product Name is automatically entered. The Name is used to refer to the product in CLI commands. See [apic products](#).
  - Click Next.
- To publish the API by adding it to an existing Product:
  - Select Existing Product - Publish using an existing product
  - Select the Product you want to use.

- Click Next.
5. Select the Catalog to which you want to publish the Product.
  6. If Spaces have been enabled in the selected Catalog, select the Space that you require.
 

Note: The Catalogs that you can select from are those that are defined for the management server and provider organization that you are connected to. If you are using the API Manager user interface, the connection details are determined by the API Manager URL that you open, and the user ID with which you log in. If you are using the API Designer user interface, you provide the management server details and user ID in the login window that opens when you first launch API Designer; see [Logging into API Connect Designer](#).

For details of how to create a Catalog in a provider organization, see [Creating and configuring Catalogs](#).
  7. If you want to publish the Product only to selected gateway services, select Publish to specific gateway services, then select the required gateway services. Only the gateway services whose type matches the gateway type setting for the Product are listed. For information on gateway types, see [API Connect gateway types](#).
  8. Click Publish.

## Results

Your Product is published to a Catalog. You can view the state of the Product in the Catalog in API Manager. If you published the product from API Designer, ensure you are logged into API Manager with the same user name and password that you used for API Designer. Click  Manage in the API Manager UI, then select the required Catalog. The Product is shown with a state of Published.

For information about the lifecycle of a product, see [The Product lifecycle](#).

If approval is required to publish Products in the Catalog, a publish approval request is sent, and the Product moves to the Pending state; the Product is published in the Catalog when the request is approved. If approval is not required, the Product is published immediately.

Note: If approval is required to stage Products to the Catalog, a stage approval request is sent. When the request is approved, the Product moves to the staged state and must be separately published in the Catalog. For information on publishing a staged Product in a Catalog, see [Publishing a Product](#). For information on configuring Product lifecycle approvals for a Catalog, see [Creating and configuring Catalogs](#). For information on approving requests, see [Approving Product lifecycle and subscription requests](#).

If the Product contains no Plans, a Plan called Default Plan is added automatically to the Product in the Catalog.

Notes:

- If an OpenAPI 3 API contains a response wildcard (which is not supported), publishing is disabled for that API. You must correct the problem before you can publish the API.
- If the Product contains an API that includes a user-defined policy that has not been imported into this Catalog, the publish will fail. The user-defined policy must be available in the required Catalog for the Product to be published successfully. For detailed instructions on how to import a user-defined policy into a Catalog, see [Importing a user-defined policy into a Catalog](#).

## Creating a new version of an API definition

You can create multiple versions of an API definition and edit the versions independently by using IBM® API Connect.



### About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

Note: If you want to publish a modified API to a production Catalog, you **must** create a new version of the API. You cannot publish an API to a production Catalog if there is already a published API with the same name and version.

### Procedure

To create a new version of an API definition, complete the following instructions:

1. In the navigation pane, click  Develop, then select the APIs tab.
2. You can create a new version of an API definition either from the Develop page, or from within the API definition itself.
  - a. To create a new version of an API definition from the Develop page, complete the following steps:
    - i. Alongside the API version that you want to work with, click the options icon , then click Save as a new version. The Save API as a new version window opens.
    - ii. Enter your new version number, then click Save.
  - b. To create a new version of an API definition from within the API definition itself, complete the following steps:
    - i. Click the title of the API definition that you want to create a new version of.
    - ii. Click the options menu icon:



- iii. Click New Version, enter your new version number, then click Submit.

Note: The version corresponds to the `info.version` property value of the API's OpenAPI definition. The `version.release.modification` version numbering scheme is recommended, for example 1.0.0.

## Results

You have created a new version of your API definition, which you can now edit independently of other versions. Each version of the API definition is listed separately on the APIs and Products page.

## Related tasks

---

- [Creating an API definition](#)

## Updating an API definition from a file

---

You can update an API by uploading a .yaml or .json file that contains the OpenAPI definition for the revised API.

### About this task

---


You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

The update operation completely replaces the previous API definition.

### Procedure

---

To update an API from a file, complete the following steps:

1. In the navigation pane, click  Develop, then select the APIs tab.
2. Click the title of the API that you want to update.
3. Click the options menu icon:



4. Click Update, then either drag and drop the file or click the link to select the file from your local file system.
5. Click Submit, then click Save.

## Updating a SOAP API

---

You can update the configuration of an existing SOAP API either by uploading a WSDL file, or by uploading a .zip archive that contains a primary WSDL file together with other WSDL or XSD files that it references.

### About this task

---

You complete this task by using the browser based API Manager UI.

You can also use the API Connect developer toolkit CLI to update a SOAP API; for more information see [API development and management commands](#).

When you initiate the update action, the specified WSDL is parsed to find a service that matches the name of the API that you are updating. If a match is found, the WSDL service is converted into a YAML file that is used to update the API. If no match is found, an error message is displayed.

Only those sections of the API that are affected by the new WSDL are replaced, the other sections are unchanged.

### Procedure

---

To update a SOAP API from a WSDL file or a .zip archive, complete the following steps:

1. In the navigation pane, click  Develop, then select the APIs tab.
2. Click the title of the SOAP API that you want to update.
3. Click the options menu icon:



4. Then click Update WSDL, and either drag and drop the file or click the link to select the file from your local file system.  
The format of the WSDL file is validated. If the format is invalid, an error message is displayed.
5. Click Next.  
The services in the selected API are displayed. If no match is found, an error message is displayed.
6. Click Upload.  
The SOAP API is updated from the specified WSDL definition.
7. Click Save to save the updates.
8. If you updated a previously published API, you must publish it again for the change to take effect:
  - a. Click the Online/Offline toggle (shown in step 3) to set the API as Offline.
  - b. Click the toggle again to set the API back to Online, which publishes the API again.

## Related tasks

---

- [Creating an API definition](#)

---

## Renaming an API definition

By renaming an API definition, you change the **name** property that, together with the version, is used to uniquely identify the API definition in API Connect. The **name** property is initially generated automatically when you first create the API definition, based on the title, but you can later change it if desired.

### About this task

---

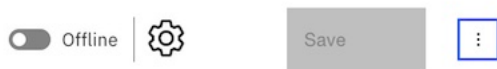
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

**API Manager UI only:** To complete this task, you must be assigned a role that has the **Api-Drafts:Edit** permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see [Creating custom roles](#).

### Procedure

---

1. In the navigation pane, click  Develop, then select the APIs tab.
2. Click the title of the API that you want to work with.
3. Click the options menu icon:



4. To rename the API, select Rename.  
Only this version of the API is renamed.
5. Enter the new name, then click Submit.  
Note: If you previously added the API to a Product, it will no longer be recognized by that Product. You must open the Product and modify the **apis**: section on the Source tab as appropriate; for example:

```
apis:
 newname1.0.0:
 name: 'newname:1.0.0'
```

---

## Changing an API rate limit

When you activate an API to make it available for testing, as described in [Activating an API](#), no rate limit is applied to restrict calls to the API. However, if desired, you can apply a rate limit to the API, or modify a previously applied rate limit. The rate limit defines the maximum number of calls allowed in a specified time period.

### About this task

---

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

**API Manager UI only:** To complete this task, you must be assigned a role that has the **Api-Drafts:Edit** permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see [Creating custom roles](#).

Note: For information about rate limits and burst limits in API Connect, see [Understanding rate limits for APIs and Plans](#).

### Procedure

---

1. In the navigation pane, click  Develop, then select the APIs tab.
2. Click the title of the API that you want to work with.
3. Click the options menu icon:



4. Click Edit Rate Limit.
5. Select Unlimited to allow unrestricted calls to the API, or select Custom Rate Limit to define the required rate limit; use the supplied fields to define the maximum number of calls allowed in a specified time period; for example, 100 calls per 1 minute.
6. Click Submit when done.

---

## Adding an OpenAPI extension to an API

You can extend the OpenAPI specification by adding either a JSON or YAML extension schema to an API. An OpenAPI extension is imported into a Catalog, or to a Space within a Catalog, then added to the API schema.

## Before you begin

---

Import the OpenAPI extension into the Sandbox Catalog, or into any Catalog to which the API that you are adding the extension to will be published; see [Importing an OpenAPI extension into a Catalog](#).

## About this task



---

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

**API Manager UI only:** To complete this task, you must be assigned a role that has the `Api-Drafts:Edit` permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see [Creating custom roles](#).

## Procedure

---

1. In the navigation pane, click  Develop, then select the APIs tab.
2. Click the title of the API that you want to work with.
3. Select the Gateway tab, expand Gateway and portal settings, then click Extensions.
4. Select the OpenAPI extensions that you want to use with this API, then click Save.  
Note: The OpenAPI extensions listed are those that have been imported into the Catalog and gateway service defined in the testing preferences for the API; see [Importing an OpenAPI extension into a Catalog](#) and [Specifying the testing preferences for an API](#). You will also need to import the OpenAPI extension into any Catalog to which the API will be published.
5. Update the OpenAPI source for the API definition with the values for the fields as defined in your selected OpenAPI extension schema files, as follows:
  - a. Click the Source icon .
  - b. Add a property whose name must begin with `x-` and references the name of the OpenAPI extension that you created. Specify the values of the properties that were defined in the OpenAPI extension YAML file.  
For an example, see [Referring to an extension in an API definition](#).
  - c. Click Save when done.

---

## Downloading an API definition

You can download the details of your API definitions so that you can store them for later recovery.

## Before you begin

---

To complete this task, you must have created an API definition. For more information, see [Creating an API definition](#).

## About this task

---

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

**API Manager UI only:** To complete this task, you must be assigned a role that has the `Drafts:View` permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see [Creating custom roles](#).

You can download REST APIs that were created by using [Creating a REST proxy API from a target service](#) for example. This download creates a .yaml file that defines the API.



You can download SOAP APIs that were created by using [Creating a REST proxy API from an existing WSDL service](#), or [Creating a SOAP proxy API from an existing WSDL service](#) for example. This download creates a .zip file that contains the WSDL definition and the YAML file that defines the API.

You can use the downloaded .yaml and .zip files to re-create your API. For information about creating an API from a .yaml file, see [Adding a REST API by importing an OpenAPI definition file](#). For information about creating an API from a .zip file, see [Adding a SOAP API by importing a zip file](#).

## Procedure

---

To download a .yaml, or .zip file that contains the details of your API, complete the following steps:

1. In the navigation pane, click  Develop, then select the APIs tab.
2. Alongside the API version that you want to work with, click the options icon  and then click Download.
3. Save the file to the required location.

## Results

---

You downloaded a .yaml, or .zip file with a representation of your API.

---

## Deleting an API definition

You can delete an API definition that is no longer required.



## About this task

---

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

**API Manager UI only:** To complete this task, you must be assigned a role that has the `Api-Drafts:Edit` permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see [Creating custom roles](#).

## Procedure

1. In the navigation pane, click  Develop, then select the APIs tab.
2. You can delete an API either from the Develop listing page, or from within the API definition itself.
  - a. To delete an API from the Develop listing page, click the options menu icon  alongside the required API, and then select Delete.
  - b. To delete an API from within the API definition, complete the following steps:
    - i. Click the title of the API definition that you want to work with.
    - ii. Click the options menu icon:



- iii. To delete just this version of the API, select Delete. To delete all versions of this API, select Delete API.

Note: If a Product was previously generated automatically from the API, by using the Activate API option, that Product will be deleted together with the API; such Products have the title `api_title` auto product.

3. Click Delete to confirm.

## Using an options file when importing a WSDL service

When you create an API definition, or add a target WSDL service to an API definition, by importing a .zip file, you can specify additional directives by including an options file in the .zip file.

You can use an options file when importing a .zip file while performing any of the following tasks:

- Creating a REST proxy API; see [Creating a REST proxy API from an existing WSDL service](#).
- Creating a SOAP proxy API; see [Creating a SOAP proxy API from an existing WSDL service](#).
- Adding a target WSDL service to an API definition; see [Editing an API definition](#).

The options file is a YAML file, and must have the file name `apiconnect.yaml`.

You can include the following fields in the file:

Table 1. Fields that can be included in the `apiconnect.yaml` file

Field name	Value	Default	Description
<code>suppressExamples</code>	<code>true</code> or <code>false</code>	<code>false</code>	Suppress auto-generation of examples.
<code>wssecurity</code>	<code>true</code> or <code>false</code>	<code>true</code>	Enable generation of definitions for WS-Security headers.
<code>implicitHeaderFiles</code>	Array of XSD file locations		Additional schema files, not referenced in the WSDL, that are used to define additional SOAP headers.
<code>port</code>	The <code>name</code> attribute for a <code>wsdl:port</code> in a <code>wsdl:service</code> definition.		Create the API by using the information for the specified port. If the specified port does not exist, or refers to a REST/XML port rather than a SOAP port, the API creation fails.  If no <code>port</code> field is present in the options file, API Connect creates the API by using the first SOAP port that it finds in the <code>wsdl:service</code> definition.

## Example

```
suppressExamples: true
wssecurity: false
implicitHeaderFiles:
- xsdDir/schema1.xsd
- xsdDir/schema2.xsd
```

## Variable references in API Connect

In API Connect you can reference different variables in your API definition.

When defining an API, creating a custom policy, or configuring another policy or logic construct, you can include references to context variables and properties.

Variable references are resolved either when the API is staged in a Product, for static variables that are fixed upon staging, or when the API is called, for variables that can change with each API call.

## Types of variables

Context variables

A context variable is a variable relevant during an API call, for example, the input of the call, the path of the call, or the message during the call. A context variable is one of the variables that makes up that particular context.

Context variables can consist of more than one part, for example, `request.headers`.



For a list of available context variables, see [API Connect context variables](#).

#### API properties

An API property is a variable in an API where its value depends upon the Catalog in which the API is staged or published. By referencing an API property, you can use the same API definition in different Catalogs where there are small differences between the instances of the API between the Catalogs. For example, an assembly could contain an `if` construct that executes its case when a particular Catalog is used, determined from the value of the API property. API properties can also be used to hide a value such as a password by encoding the value.

API properties are referenced by name.

For a list of API properties, see [API properties](#)

For more information, see [Setting API properties](#) (OpenAPI 2.0) or [Setting API properties](#) (OpenAPI 3.0).

Note: Once defined, an API property is read only.

#### Catalog properties

A Catalog property is specific to a Catalog, and can be referenced in any of the API definitions in that Catalog. For more information, see [Creating and configuring Catalogs](#).

#### Notes:

- If you change the value of a Catalog property, any API that references that property must be republished for it use the new value.
- Catalog properties and API properties are not supported with global policies. Therefore, if you use such properties in a global policy, they are not replaced with the values specified in the Catalog or API property definitions.

## Methods of referencing variables

You can get, or set, the value for a referenced variable.

- Get the value for a variable in either of the following ways:
  - Through the [GatewayScript](#) policy, run the `apim.getvariable()` API.
  - **API Gateway only** Through the [GatewayScript](#) policy, use the `context.get()` function; for more information, see [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway](#).
  - Through the [XSLT](#) policy, run a stylesheet that uses the `apim:getVariable` extension function.
  - In an assembly policy field that supports variable references, use the following syntax:  
  
`$(variable)`
- Set the value for a variable in either of the following ways:
  - Through the [Set Variable](#) policy.
  - Through the [GatewayScript](#) policy, use the `apim.setvariable()` API.
  - **API Gateway only** Through the [GatewayScript](#) policy, use the `context.set()` function; for more information, see [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway](#).
  - Through the [XSLT](#) policy, run a stylesheet that uses the `apim:setVariable` extension element.

#### GatewayScript references

When you want to reference a variable in a GatewayScript context, use one of the following methods:

```
apim.getvariable(variable)
```

where *variable* is the name of the context variable or API property that you want to reference.

```
apim.setvariable(variable, value, action)
```

where

- *variable* is the name of the context variable or API property that you want to reference.
- *value* is the string value that you want to set the variable to. This can be a literal value, or another variable. For example, to set your named variable to the value of the Content-Type header in a request, use the following code:

```
var contentType = apim.getvariable('request.headers.content-type');
apim.setvariable(variable, contentType, 'set');
```

This property is required only when `set` or `add` is specified as the action.

- *action* is the action that you want to apply to the variable. Valid options are:
  - `set`
  - `add`
  - `clear`

If no option is set, the default option of `set` is applied.

Use the `getvariable` method to retrieve the value of a context variable or API property, and the `setvariable` method to change one.

Some of the situations where you would use this type of reference are:

- GatewayScript policies. -For more information, see [GatewayScript](#).
- `if` logic constructs. For more information, see [if](#).
- User-defined policies. For more information, see [Authoring policies](#).

#### Stylesheet references

You can reference a variable by using functions and elements in an XSLT policy with the following syntax:

```
<xsl:variable name="variable_name" select="apim:getVariable(variable)" />
```

where *variable* is a literal value, another variable, or a valid XSLT XPath statement.

```
<xsl:call-template name="apim:setVariable">
 <xsl:with-param name="varName" select="variable"/>
 <xsl:with-param name="value" select="value"/>
 <xsl:with-param name="action" select="action"/>
</xsl:call-template>
```

where

- *variable* is the name of the context variable or API property that you want to reference. This can be a literal value, another variable, or a valid XSLT XPath statement.
- *value* is the string value that you want to set the variable to. This can be a literal value, another variable, or a valid XSLT XPath statement. This property is required only when **set** or **add** is specified as the action.
- *action* is the action that you want to apply to the variable. This can be a literal value, another variable, or a valid XSLT XPath statement. Valid options are:
  - **set**
  - **add**
  - **clear**

If no option is set, the default option of **set** is applied.

The following example sets a named variable to the value of the Content-Type header in a request:

```
<xsl:variable name="contentType" select="apim:getVariable('request.headers.content-type') " />
<xsl:call-template name="apim:setVariable">
 <xsl:with-param name="varName" select="'variable'"/>
 <xsl:with-param name="value" select="$contentType"/>
 <xsl:with-param name="action" select="'set'"/>
</xsl:call-template>
```

Inline references

In many situations you can make a simpler reference, by using the following syntax:

```
$(variable)
```

where *variable* is the name of the context variable or API property that you want to reference.

Some of the situations where you would use this type of reference are:

- The URL called by an **Invoke** or **Proxy** policy. For more information about the policies, see [Invoke](#) or [Proxy](#).
- A **Map** policy. For more information, see [Map](#).

Notes: The map policy can reference the following variables inline:

- Variables that are defined as inputs to the map policy and specified in the from field of a mapping.
- Context variable or API properties, provided that the `x-ibm-gateway-map-resolve-apic-variables` API property is not set to **false**. If an inline reference in a map policy resolves to a context variable or API property, it is immediately replaced by the corresponding value. For more information on the `x-ibm-gateway-map-resolve-apic-variables` API property, see [Properties controlling the map policy](#).

For more information on the use of inline references, and examples, see the following [About inline references](#) section.

## About inline references

When you use an inline reference for a property value in an assembly policy, the variable reference is replaced with the corresponding string value at run time, before the property is evaluated. For example, consider the following incoming request, which has two headers and two query parameters (a **curl** command is used to illustrate the request):

```
curl --data-binary @request.json \
 -H 'Content-Type: application/json' \
 -H 'X-Environment: test' \
 http://apiserver/org/catalog/petstore/getPetDetails?petid=285&storeid=z03
```

If an **invoke** policy is configured as follows:

```
invoke:
 target-url: https://backend/GetPetInfo/$(request.parameters.storeid)/$(request.parameters.petid)
```

then the value of the `target-url` property is evaluated as `https://backend/GetPetInfo/z03/285`, and this is therefore the URL to which the **invoke** policy sends its request.

**DataPower Gateway (v5 compatible) only**

You can also use an inline reference in the `condition` property of a **switch** policy. Consider the following example:

```
- switch:
 case:
 - condition: "$(request.headers.X-Environment)" == "production"
 execute:
 - invoke:
 http://www.finance.com/base/stockquote
 - condition: "$(request.headers.X-Environment)" == "test"
 execute:
 - invoke:
 http://testserver1.finance.com/stockquote
```

Using the incoming request referred to previously, the value of the `X-Environment` header is `test`. The inline references are first evaluated, then the conditions are evaluated as JavaScript. Therefore, the first condition is evaluated as `test == production`, returning **false**, and the second condition is evaluated as `test == test`, returning **true**, so the **invoke** policy sends its request to `http://testserver1.finance.com/stockquote`.

Important: This example, which encloses the inline references inside double quotes, assumes that the API is deployed to the DataPower API Gateway. If you deploy the API to the DataPower Gateway (v5 compatible), then the double quotes are added implicitly when the inline references are evaluated, and you must therefore omit them from the `condition` property; the two conditions in this example would therefore be

```
condition: $(request.headers.X-Environment) == "production"
```

and

```
condition: $(request.headers.X-Environment) == "test"
```

However, as an alternative to using an inline reference in a `condition`, you can instead use pure JavaScript, as follows:

```
- switch:
 case:
 - condition: request.headers["X-Environment"] == "production"
 execute:
 - invoke:
 http://www.finance.com/base/stockquote
 - condition: request.headers["X-Environment"] == "test"
 execute:
 - invoke:
 http://testserver1.finance.com/stockquote
```

in which case the policy configuration will work correctly as-is in both types of gateway. For more information on the types of gateway, see [API Connect gateway types](#).

In addition to single strings, you can use an inline reference for a complete object. Consider the following example, which uses an inline reference in a `set-variable` policy:

```
- set-variable:
 actions:
 - set: saved.reqHdrs
 value: $(request.headers)
 type: string
```

Here, the `request.headers` structured JSON object is substituted with its corresponding string representation, and a variable named `reqHdrs` is therefore created, under the `saved` object, with a string value of `{ "Content-Type": "application/json", "X-Environment": "test", ... }`. However, if you are using the DataPower API Gateway, you can specify `type: any`, in which case the `set-variable` policy copies the value as-is, and `saved.reqHdrs` is created as a JSON object rather than a string:

```
- set-variable:
 actions:
 - set: saved.reqHdrs
 value: $(request.headers)
 type: any
```

You can reference a property in a JSON or XML request directly by name, by using the following syntax:

```
$(request.body.property_name)
```

For example, if the request contains `{ "account-number": 123 }` or `<account-number>123</account-number>` you can retrieve the value `123` by using the following inline reference:

```
$(request.body.account-number)
```

If the property value is within a nested property structure, you can reference it by concatenating the property names. For example, if the request contains `{ "account": { "balance": 123 } }` or `<account><balance>123</balance></account>` you can retrieve the value `123` by using the following inline reference:

```
$(request.body.account.balance)
```

Similarly, you can reference a property in a JSON or XML response by using the following syntax:

```
$(message.body.property_name)
```

Restriction: If you are using the DataPower API Gateway, you can reference properties in this way only if the format of the request or response is JSON; the mechanism is not supported with XML request or response formats. If you are using the DataPower Gateway (v5 compatible) the mechanism is supported with both JSON and XML formats.

## API properties

Properties are used by the gateway to control behavior of certain policies. Typically, you provide properties, but the policy can also provide properties settings.

In IBM® API Connect, you can create API properties that consist of Catalog-specific values to eliminate the need for source code modifications. You can then reference the properties elsewhere in your API definition.

Pre-supplied API properties for various policies are shown in the tables.

A list of invoke related API properties that control the behavior of the invoke policy.

Table 1. Properties controlling the invoke policy

Property	Required	Description	Data type
<a href="#">IBM® API Connect v10.0.5.x LTS</a> x-ibm-gateway-decode-request-params	No	If set to a value of <code>true</code> , any request parameters that are referenced by a variable definition on an invoke target-url are URL-decoded. The default behavior is to not decode any parameters, thereby sending them to the target URL without alteration.	Boolean

Property	Required	Description	Data type
<small>DataPower Gateway (v5 compatible only)</small> x-ibm-gateway- invoke- suppress- clientid	No	When set with a value of <b>true</b> , or not specified, the X-IBM-Client-Id HTTP header (if specified on the API request) is suppressed from being sent to the invoke target URL. When set with a value of <b>false</b> , the X-IBM-Client-Id HTTP header is no longer suppressed from being sent to the invoke target URL. This property is supported only by the DataPower® Gateway (v5 compatible). If you are using the DataPower API Gateway then to achieve the same functionality add a <b>header-control</b> property to the <b>invoke</b> policy configuration in the OpenAPI definition for your API, as in the following examples.  Suppress the <b>X-Client-ID</b> header as follows: <pre>- invoke:   target-url: http://myhostname/mypath   header-control:     type: blacklist     values:       - ^X-Client-ID\$</pre> Suppress the <b>userId</b> query parameter as follows: <pre>- invoke:   target-url: http://myhostname/mypath   parameter-control:     type: blacklist     values:       - ^userId\$</pre>	Boolean
<small>DataPower Gateway (v5 compatible only)</small> x-ibm-gateway- optimize-invoke	No	If set to <b>false</b> , prevents the replacement of the last invoke in a policy with proxy. Any value other than <b>false</b> (case insensitive) will result in the last invoke in a policy possibly being replaced by proxy when the API is executed in the gateway.	Boolean
<small>DataPower Gateway (v5 compatible only)</small> x-ibm-gateway- queryparam- encode-plus- char	No	If set to a value of <b>true</b> , all "+" characters in the query parameter values of the target-url of the Invoke and Proxy policies are encoded to "%2F". The default value is <b>false</b> .	Boolean
<small>DataPower Gateway (v5 compatible only)</small> x-ibm-gateway- api-enforce- response-limits	No	If set to a value of <b>true</b> , allows the JSON parser to be enforced on the response rule. If the response body size is higher than the JSON parser limit set in the DataPower domain, a status code of 500 is returned. Note: The x-ibm-gateway-api-enforce-response-limits property is supported by the DataPower Gateway (v5 compatible) but not by the DataPower API Gateway. However, if you are using the DataPower API Gateway, consider using a <a href="#">Parse</a> policy in your API assembly to enforce these limits. For information on the different types of gateway, see <a href="#">API Connect gateway types</a> .	Boolean
<small>DataPower Gateway (v5 compatible only)</small> x-ibm-gateway- invoke-emulate- v4-soap-error	No	IBM API Management Version 4.0 initiates a DataPower error when a SOAP fault is returned from a web service. IBM API Connect provides a mechanism to catch SOAP errors and does not initiate a DataPower error. For compatibility with APIs developed in IBM API Management Version 4.0, set this property to <b>true</b> only in the case where a gateway extension is expecting to handle a SOAP error in a post error rule. The default value is <b>false</b> .  Note: This property is deprecated in favor of x-ibm-gateway-invoke-emulate-v4-invoke-error.	Boolean
<small>DataPower Gateway (v5 compatible only)</small> x-ibm-gateway- invoke-keep- payload	No	If set to a value of <b>true</b> , the invoke policy sends a payload on an HTTP DELETE method. This property is available for use with IBM DataPower Gateway version 7.7.1.1 and later. The default value is <b>false</b> .	Boolean
<small>DataPower Gateway (v5 compatible only)</small> x-ibm-gateway- invoke-emulate- v4-invoke-error	No	IBM API Management Version 4.0 initiates a DataPower error when a backend server error is returned, either a SOAP fault returned from a web service or a JSON or XML (non-SOAP) error from a restful service. IBM API Connect provides a mechanism to catch SOAP errors and operation errors and does not initiate a DataPower error when they occur. If no catch policy is configured, a generic error message is generated. For compatibility with APIs developed in IBM API Management Version 4.0, set this property to <b>true</b> only in the case where a gateway extension is expecting to handle a backend server error in a gateway extension post error rule or if the client of the API is expecting the backend server error to be returned. The default value is <b>false</b> .	Boolean

A list of map-related API properties that control the behavior of the map policy.

Table 2. Properties controlling the map policy

Property	Required	Description	Data type
<small>DataPower Gateway (v5 compatible only)</small> x-ibm-gateway- map-array-first- element-value	No	In IBM API Management Version 4.0, if a mapping source value is from an array then only the first value is output. In API Connect, the default behavior is to return an array of all array element values. To maintain compatibility with IBM API Management Version 4.0, set this API property to <b>true</b> to only return the first array element value.	Boolean
<small>DataPower Gateway (v5 compatible only)</small> x-ibm-gateway- map-resolve- apic-variables	No	By default, any API Connect variable that is found in the map configuration is resolved. For example, <b>\$(request.headers.content-type)</b> resolves to the request's content type header. Because searching for variables in every map property can be CPU intensive, you can choose not to resolve variables by setting this API property to <b>false</b> . If this property is not configured or is set to any other value, the existing behavior to search for these variables continues. Note that variable usage within a map value JavaScript snippet is not changed provided that the variables that are referenced come from a configured map input.	Boolean
<small>DataPower Gateway (v5 compatible only)</small> x-ibm-gateway- map-create- empty-array	No	This property controls how the map policy handles the output of an empty array; it can have the following values: <ul style="list-style-type: none"> <li><b>all</b>: Output all empty arrays, including empty children arrays. This is the default value if the property is not configured or has an invalid value.</li> <li><b>parent</b>: Output only the current property's empty array value. Children map actions of this property are not attempted.</li> <li><b>none</b>: Prevent any empty output array values from being produced.</li> </ul>	String

Property	Required	Description	Data type
<small>DataPower Gateway (5.5 compatible only)</small> x-ibm-gateway-optimize-schema-definition	No	Set the value of this property to <b>true</b> to provide a performance improvement to the map policy when a very complex schema definition is referenced by a policy output definition; for example, some very complex schemas that are generated by importing a very complex WSDL schema. The map policy builds a schema from an API definition when a referenced definition is provided as the value of the schema. If the schema does not have references that generate a circular reference, setting this property to <b>true</b> might provide a performance benefit while generating the same schema as would otherwise have been generated. However, in cases where the schema is very complex, with many potentially circular references, the generated schema could be different because the enhanced schema handling processes circular references differently. In such cases therefore, you should examine the resulting output to determine if the performance benefit gained is not at the expense of a change in the map policy output.  The default value of this property, is <b>false</b> , maintaining the existing behavior and performance.	Boolean
<small>DataPower Gateway (5.5 compatible only)</small> x-ibm-gateway-map-null-value	No	Set the value of this API property to <b>true</b> to allow a property from a map policy's input data with a value of <b>null</b> to be mapped to the output document. By default, a property from a map policy's input data with a value of <b>null</b> is not mapped to the output document.	Boolean
<small>DataPower Gateway (5.5 compatible only)</small> x-ibm-gateway-map-resolve-xmlinput-datatypes	No	XML input elements with numeric or boolean data have no metadata to indicate whether this data should be mapped as a string value or as the specific data type. If you set the value of this property to <b>false</b> , XML input elements are always mapped as a string. If you set the value to <b>true</b> , numeric or boolean XML input elements are mapped as the corresponding data type from the input schema. The default value is <b>false</b> .	Boolean
<small>DataPower Gateway (5.5 compatible only)</small> x-ibm-gateway-map-xml-empty-element	No	This property controls how the map policy handles XML input empty elements and impacts JSON output when the input document is XML; it can have the following values: <ul style="list-style-type: none"> <li>• <b>string</b>: The value for an empty XML element is considered to be an empty string. This is the default value if the property is not configured or has an invalid value.</li> <li>• <b>null</b>: The value for an empty XML element is considered to be null. A mapping of this element to a JSON output property does not occur unless the API property x-ibm-gateway-map-null-value is also specified with a value of <b>true</b>.</li> <li>• <b>none</b>: The empty XML element is ignored.</li> <li>• <b>string-badgerfish</b>: The value for an empty XML element is considered to be an empty string. The empty string value will be placed into a JSON badgerfish value property.</li> <li>• <b>null-badgerfish</b>: The value for an empty XML element is considered to be null. The null value will be placed into a JSON badgerfish value property. A mapping of this element to a JSON output property does not occur unless the API property x-ibm-gateway-map-null-value is also specified with a value of <b>true</b>.</li> </ul>	Boolean
<small>DataPower Gateway (5.5 compatible only)</small> x-ibm-gateway-schema-definition-reference-limit	No	Set the value of this property to an integer value that specifies the maximum allowed number of iterations of a circular schema definition. The default value is 1, which means that circular schema definitions are not followed. The maximum possible value is 5. If you specify a value greater than 5, a value of 5 is assumed. If you specify a non-numeric value, a value of 1 is assumed.	String

Property	Required	Description	Data type
<small>DataPower Gateway (v5 compatible) only</small> x-ibm-gateway-map-emulate-v4-default-required-properties	No	<p>Set this property to true to have default values generated in the output for required properties that are either not mapped, or for which there is no input data present, in the following specific cases:</p> <ul style="list-style-type: none"> <li>An array consists of objects that contain one or more required properties.</li> <li>An object which is optional has one or more child properties that are required.</li> </ul> <p>By default, these required properties are not present in the output. If you set the <b>x-ibm-gateway-map-emulate-v4-default-required-properties</b> API property to <b>true</b>, these required properties will be present in the output. If the output schema defines a <b>default</b> property for the output property then the specified default value is used, otherwise a default value is assigned dependent on the data type, as follows:</p> <ul style="list-style-type: none"> <li>String: empty string ("")</li> <li>Number: 0</li> <li>Boolean: false</li> <li>Object: empty object</li> <li>Array: empty array</li> </ul> <p><b>Example 1</b> The input data has the following array of objects:</p> <pre>[{"a": "value1"}, {"a": "value2"}, {"b": "value3"}]</pre> <p>The output schema defines the output object as having two properties, <b>a</b> and <b>b</b>, of which <b>b</b> is required. The map policy defines the following mappings:</p> <ul style="list-style-type: none"> <li><b>input.array.a</b> to <b>output.array.a</b></li> <li><b>input.array.b</b> to <b>output.array.b</b></li> </ul> <p>If the <b>x-ibm-gateway-map-emulate-v4-default-required-properties</b> API property is set to <b>true</b>, and <b>b</b> is either not mapped or has no input data present, then <b>b</b> is assigned a default value of an empty string, and the output is as follows:</p> <pre>[{"a": "value1", "b": ""}, {"a": "value2", "b": "value3"}]</pre> <p><b>Example 2</b> The output schema defines the following structure:</p> <pre>{"a" : {"b" : {"c" : "value1", "d" : "value2"} } }</pre> <p>Property <b>b</b> is optional but property <b>d</b> within <b>b</b> is required.</p> <p>The map policy defines a mapping to <b>output.a.b.c</b>.</p> <p>If the <b>x-ibm-gateway-map-emulate-v4-default-required-properties</b> API property is set to <b>true</b>, and <b>d</b> is not mapped, then <b>d</b> is assigned a default value of an empty string, and the output is as follows:</p> <pre>{"a" : {"b" : {"c" : "value1", "d" : ""} } }</pre> <p>If the <b>x-ibm-gateway-map-emulate-v4-default-required-properties</b> API property is not specified or does not have a value of <b>true</b>, these required properties are <b>not</b> created in the output with their default values.</p>	Boolean
<small>DataPower Gateway (v5 compatible) only</small> x-ibm-gateway-map-post-process-json-output	No	<p>Set the value of this property to <b>true</b> to enable post processing of mapped JSON output. The post processing of JSON output will use the output schema to ensure that property values are of the same data type as that defined in the schema. It will also normalize output property values that have a Badgerfish JSON syntax due to object mapping of an XML input. Set the value to <b>false</b> for no post processing of mapped JSON output. The default value is <b>false</b>.</p>	Boolean
<small>DataPower Gateway (v5 compatible) only</small> x-ibm-gateway-map-emulate-v4-empty-json-object	No	<p>If a mapping fails because its input is not present and there is no default mapping configured, the default behavior is to not to make any change to the output mapping. Set the value of this property to <b>true</b> to create an empty object for the parent of the target mapping, emulating the behavior of IBM API Management Version 4.0.</p> <p><b>Example</b> The map policy defines a mapping to <b>output.a.b.c</b>. If input data is present, the output is as follows:</p> <pre>{   "a": {     "b": {       "c": "inputvalue"     }   } }</pre> <p>If there is no input data, and the <b>x-ibm-gateway-map-emulate-v4-empty-json-object</b> API property is set to <b>true</b>, the output is as follows:</p> <pre>{   "a": {     "b": {     }   } }</pre> <p>Properties <b>a</b> and <b>b</b> are created but the value of <b>b</b> is an empty object. The default value is <b>false</b>.</p>	Boolean

DataPower Gateway (v5 compatible) only

A list of proxy-related API properties that control the behavior of the proxy policy.

Table 3. Properties controlling the proxy policy

Property	Required	Description	Data type
x-ibm-gateway-proxy-suppress-clientid	No	<p>A setting of <code>false</code> activates the injection of the X-IBM-Client-Id HTTP header (if it is specified on the API request), or the <code>client_id</code> query parameter in the request URL, to the proxy target-url. If not specified, or set with a value of <code>true</code>, suppresses the sending of this parameter to the proxy target-url.</p> <p>This property is supported only by the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway then to achieve the same functionality add a <code>header-control</code> property to the <code>invoke</code> policy configuration in the OpenAPI definition for your API, as in the following examples.</p> <p>Suppress the <code>X-Client-ID</code> header as follows:</p> <pre>- proxy:   target-url: http://myhostname/mypath   header-control:     type: blacklist     values:       - ^X-Client-ID\$</pre> <p>Suppress the <code>userId</code> query parameter as follows:</p> <pre>- proxy:   target-url: http://myhostname/mypath   parameter-control:     type: blacklist     values:       - ^userId\$</pre>	Boolean
x-ibm-gateway-optimize-invoke	No	If set to <code>false</code> , prevents the replacement of the last invoke in a policy with proxy. Any value other than <code>false</code> (case insensitive) will result in the last invoke in a policy possibly being replaced by proxy when the API is executed in the gateway.	Boolean
x-ibm-gateway-queryparam-encode-plus-char	No	If set to a value of <code>true</code> , all "+" characters in the query parameter values of the target-url of Invoke and Proxy policies are encoded to "%2F". The default value is <code>false</code> .	Boolean
x-ibm-gateway-api-enforce-response-limits	No	If set to a value of <code>true</code> , allows the JSON parser to be enforced on the response rule. If the response body size is higher than the JSON parser limit set in the DataPower domain, a status code of 500 is returned.	Boolean

A list of API properties that control the behavior of the rate limit policy.

Table 4. Properties controlling the rate limit policy

Property	Required	Description	Data type
x-ibm-gateway-emulate-v4-plan-rate-limit	No	By default in IBM API Connect Version 10, if you configure a rate limit only for a Plan and not for the API operations within the Plan then a single rate limit threshold is set for the API as a whole, regardless of which operation in the API is requested. This behavior differs from IBM API Management Version 4.0 where the rate limit is set <b>individually</b> for <b>each</b> operation in the API. To change the Version 10 behavior to emulate the Version 4.0 behavior, set this API property to a value of <code>true</code> .	Boolean

A list of API properties that control the behavior of multiple policies.

Table 5. Properties controlling multiple policies

Property	Required	Description	Data type
<small>DataPower Gateway (v5 compatible) only</small> x-ibm-gateway-sourcecode-resolve-apic-variables	No	<p>If set to <code>true</code>, API Connect variable references are resolved. Set to <code>false</code> if you want the policy to ignore API Connect variable references.</p> <p>The default value is <code>true</code>.</p> <p>This property applies to the following policies:</p> <ul style="list-style-type: none"> <li>Map</li> <li>GatewayScript</li> <li>XSLT</li> <li>if</li> <li>switch</li> </ul> <p>Note: This property setting is overridden by the x-ibm-gateway-map-resolve-apic-variables API property setting for the Map policy.</p>	Boolean
<small>DataPower Gateway (v5 compatible) only</small> x-ibm-gateway-api-json-parse-error-handling	No	<p>If an API request or response payload includes valid JSON content that contains characters that cannot be represented in the JSON XML internal syntax that is used by the DataPower Gateway, set this property to <code>escape-unicode</code> to allow the payload to be accepted without parsing errors. If this property is not configured or is set to any other value, the payload is rejected as invalid JSON.</p> <p>This property applies to the API request payload, and to the API response payload when <code>x-ibm-gateway-api-enforce-response-limits</code> is enabled.</p>	String
<small>DataPower Gateway (v5 compatible) only</small> x-ibm-gateway-framework-preserve-escaped-reverse-solidus	No	By default, the string <code>\\</code> in a policy property is converted to a single <code>\</code> character. Set this property to <code>true</code> to preserve the string <code>\\</code> .	Boolean

Property	Required	Description	Data type
<small>DataPower Gateway (v5 compatible) only</small> x-ibm-gateway-inspect-request-headers	No	Causes an inspection of the HTTP headers in the API request to check for characters in the header values that are illegal XML characters; it can have the following values: <ul style="list-style-type: none"> <li><b>default:</b> There is no inspection for these characters in the header values. If one is present, the API request fails with an HTTP 500 Internal Server Error.</li> <li><b>sanitize:</b> Any illegal XML characters in header values are replaced with a ? character. The API processing will continue. Any API that attempts to read <code>request.headers.&lt;headername&gt;</code> will see the ? character in the value. However, the original protocol headers representing <code>message.headers</code> will still have the original character, which will be sent to an invoke or proxy backend server.</li> <li><b>bad-request:</b> There will be an inspection for these characters in the header values. If one is present, the API request fails with an HTTP 400 Bad Request.</li> </ul> <p>The default value is <b>default</b>.</p>	Boolean

A list of API properties that control the behavior of custom policies.

Table 6. Properties controlling custom policies

Property	Required	Description	Data type
<small>DataPower Gateway (v5 compatible) only</small> x-ibm-gateway-custom-policy-with-gws-action	No	If set to <b>true</b> , the <code>request.body</code> and <code>message.body</code> context variables will be populated for access by an <code>apim.getvariable('request.body')</code> or <code>apim.getvariable('message.body')</code> function call in a GatewayScript action of a custom policy. If the custom policy does not use a GatewayScript action that requires these variables to be populated, set this property to <b>false</b> or do not specify it. The default value is <b>false</b> .	Boolean

## API policies and logic constructs

Policies and logic constructs are pieces of configuration that control a specific aspect of processing in the Gateway server during the handling of an API invocation at run time.

Policies are the building blocks of assembly flows, and they provide the means to configure capability, such as security, logging, routing of requests to target services, and transformation of data from one format to another. Policies can be configured in the context of an API or in the context of a Plan.

Logic constructs behave in a similar way to policies, but they affect how and which parts of the assembly are implemented without modifying the data flow of the assembly.

IBM® API Connect provides the following ways that you can create, configure, and apply policies and logic constructs:

### Policies associated with a Plan

A Plan provides a mechanism for grouping API operations or subsets of operations from one or more APIs. You can set rate limiting policies on a Plan to specify how many requests an application is allowed to make during a specified time interval. You can also configure a policy for each operation that is included in a Plan. For more information, see [Working with Products](#).

### Built-in policies

A built-in policy enables you to apply a pre-configured policy statement to an assembly to control processing capabilities in the Gateway server. Built-in policies are applied by using the API Designer assembly editor to add a built-in policy to your assembly and to configure the properties for that policy. For more information, see [Built-in policies](#).

Note: You can also apply built-in policies to your APIs by adding an **assembly** extension to your OpenAPI definition file. For more information, see [IBM extensions to the OpenAPI specification](#).

### Logic constructs

A logic construct enables you to control the flow of data through your assembly during an API call. Like policies, logic constructs are applied to an API by using the API Designer assembly editor to add a logic construct to your assembly and to configure the behavior of the construct. For more information, see [Logic Constructs](#).

Note: You can also apply logic constructs to your APIs by adding an **assembly** extension to your OpenAPI definition file. For more information, see [IBM extensions to the OpenAPI specification](#).

### User-defined policies

A user-defined policy enables you to create your own policies to control extra processing features in the Gateway server, such as security, or routing of requests. User-defined policies are created outside of API Connect and then imported into one or more Catalogs, so they can be applied to an operation in the same way as built-in policies. For more information, see [Authoring policies](#).

- [Built-in policies](#)

IBM API Connect includes a number of built-in policies that you can use to apply preconfigured policy statements to an operation to control an aspect of processing in the Gateway server when an API is invoked.

- [Logic Constructs](#)

IBM API Connect includes a number of logic constructs that you can use to apply preconfigured logic to an assembly to control the flow of data through your assembly when the API is called.

- [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway](#)

In the code for GatewayScript and XSLT policies in your API assemblies, you can use API context variables to work with the messages that are generated when an API is called.

- [User-defined policies](#)

Make a user-defined policy available to a Catalog, and apply that policy to a REST or SOAP API.

## Related reference



- [execute](#)

## Built-in policies



IBM® API Connect includes a number of built-in policies that you can use to apply preconfigured policy statements to an operation to control an aspect of processing in the Gateway server when an API is invoked.

Note: Although some built-in policies can be used with both the DataPower® Gateway (v5 compatible) and the DataPower API Gateway, some policies are restricted to a particular Gateway. The following icons indicate which Gateway each policy can be used with:

-  Indicates that the policy can be run on the DataPower Gateway (v5 compatible).
-  Indicates that the policy can be run on the DataPower API Gateway.


For details of the two types of gateway, see [API Connect gateway types](#).

Built-in policies are configured in the context of an API. You can use the API Designer assembly editor to add a built-in policy to an API and to configure the properties for that policy.

You can also add built-in policies to an API by creating an OpenAPI definition file. For more information, see [Creating an OpenAPI definition file](#).

The following table shows the list of built-in policies that are available. The table contains links to configuration information for both the built-in policy definitions, and the OpenAPI policy definitions. The policies are the same, but they are created in different ways.

Table 1. Built-in policies

Built-in policy	OpenAPI policy	Description		
<a href="#">Activity Log</a>	<a href="#">activity-log</a>	Use the Activity Log policy to configure your logging preferences for the API activity that is stored in IBM API Connect analytics. The preferences that you specify will override the default settings for collecting and storing details of the API activity. Note: The Activity Log policy is not supported in the assembly for an API whose gateway type is DataPower API Gateway. Instead, you configure activity logging in the API design settings. For details, see <a href="#">Configuring activity logging</a> (OpenAPI 2.0) or <a href="#">Configuring activity logging</a> (OpenAPI 3.0).	✓	✓ Functionality provided in the API design; see <a href="#">Configuring activity logging</a> (OpenAPI 2.0) or <a href="#">Configuring activity logging</a> (OpenAPI 3.0)
<a href="#">Client Security</a>	<a href="#">client-security</a>	Provides a range of options for authenticating client access to your APIs, extending the capabilities of the OpenAPI specification.	✗	✓
<a href="#">GatewayScript</a>	<a href="#">gatewayscript</a>	Use the <code>gatewayscript</code> policy to execute a specified DataPower GatewayScript program.	✓	✓
<a href="#">Generate JWT</a>	<a href="#">jwt-generate</a>	Use the Generate JWT security policy in IBM API Connect to generate a JSON Web Token (JWT).	✓	✓
<a href="#">Validate JWT</a>	<a href="#">jwt-validate</a>	Use the Validate JWT security policy to enable the validation of a JSON Web Token (JWT) in a request before allowing access to the APIs.	✓	✓
<a href="#">if</a>	<a href="#">if</a>	Use the if policy to apply a section of the assembly when a condition is fulfilled.	✓	✓ Functionality provided by <a href="#">switch</a>
<a href="#">GraphQL introspect</a>	<a href="#">graphql-introspect</a>	Use the GraphQL introspect policy to introspect a GraphQL schema.	✗	✓
<a href="#">Invoke</a>	<a href="#">invoke</a>	Apply the Invoke policy to call another service from within your assembly. The response from the backend is stored either in the variable <code>message.body</code> or in the response object variable if it is defined. The policy can be used with JSON or XML data, and can be applied multiple times within your assembly.	✓	✓
<a href="#">JSON to XML</a>	<a href="#">json-to-xml</a>	Use the JSON to XML policy to convert the context payload of your API from the JavaScript Object Notation (JSON) format to the extensible markup language (XML) format.	✓	✓
<a href="#">Log</a>	<a href="#">log</a>	Use the Log policy to customize or override the default activity logging configuration for an API.	✗	✓
<a href="#">Map</a>	<a href="#">map</a>	Use the Map policy to apply transformations to your assembly flow and specify relationships between variables.	✓	✓
<a href="#">operation-switch</a>	<a href="#">operation-switch</a>	Use the operation-switch policy to apply a section of the assembly to a specific operation.	✓	✓
<a href="#">OAuth</a>	<a href="#">oauth</a>	Use the OAuth policy to policy to perform OAuth processing based on defined OAuth provider settings.	✗	✓
<a href="#">Parse</a>	<a href="#">parse</a>	Use the Parse policy to control the parsing of an input document. When the input document is a JSON string, the string is parsed instead of copied over.	✗	✓
<a href="#">Proxy</a>	<a href="#">proxy</a>	Apply the Proxy policy to invoke another API within your assembly, particularly if the separate API contains a large payload. The response from the backend is stored in the <code>message.body</code> and in the response object variable if it is defined. Only one policy is permitted to be run per unique assembly flow.	✓	✓ Functionality provided by <a href="#">Invoke</a>
<a href="#">Rate Limit</a>	<a href="#">ratelimit</a>	Use the Rate Limit policy to apply one or more rate or burst limits at any point in your API assembly flow. Rate and burst limits restrict the number of calls that an application can make to an API in a specified time period.	✗	✓
Redaction <a href="#">Redaction - DataPower API Gateway</a> <a href="#">Redaction - DataPower Gateway (v5 compatible)</a>	<a href="#">redact - DataPower API Gateway</a> <a href="#">redact - DataPower Gateway (v5 compatible)</a>	Use the Redaction policy to completely remove or to redact specified fields from the Request body, the Response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.	✓	✓
<a href="#">Set Variable</a>	<a href="#">set-variable</a>	Use the Set Variable policy to set the value of a runtime variable, or to clear a runtime variable, or to add a header variable.	✓	✓

Built-in policy	OpenAPI policy	Description	DataPower Gateway	API Gateway
<a href="#">switch</a>	<a href="#">switch</a>	Use the switch policy to execute one of a number of sections of the assembly based on which specified condition is fulfilled.	✓	✓
<a href="#">throw</a>	<a href="#">throw</a>	Use the throw policy to throw an error when it is reached during the execution of an assembly flow.	✓	✓
<a href="#">User Security</a>	<a href="#">user-security</a>	Use the user-security policy to extract a user's credentials, authenticate those credentials, and obtain authorization from the user.	✗	✓
Validate <a href="#">Validate - DataPower API Gateway</a> <a href="#">Validate - DataPower Gateway (v5 compatible)</a>	<a href="#">validate - DataPower API Gateway</a> <a href="#">validate - DataPower Gateway (v5 compatible)</a>	Use the Validate policy to validate the payload in an assembly flow against a JSON or an XML schema.	✓	✓
<a href="#">Validate Username Token</a>	<a href="#">validate-username token</a>	Use the Validate Username Token policy to validate a Web Services Security (WS-Security) UsernameToken in a SOAP payload before allowing access to the protected resource.	✓	✗
<a href="#">XML to JSON</a>	<a href="#">xml-to-json</a>	Use the XML to JSON policy to convert the context payload of your API from the extensible markup language (XML) format to JavaScript Object Notation (JSON).	✓	✓
<a href="#">XSLT</a>	<a href="#">xslt</a>	Use the XSLT policy to apply an XSLT transform to the payload of the API definition.	✓	✓

[Including elements in your OpenAPI 2.0 API assembly](#), [Including elements in your OpenAPI 3.0 API assembly](#)

- **[Activity Log](#)**  
Use the Activity Log policy to configure your logging preferences for the API activity that is stored in IBM API Connect analytics. The preferences that you specify will override the default settings for collecting and storing details of the API activity.
- **[Client Security](#)**  
The Client Security policy can be used to extend client authentication access for your APIs.
- **[Extract](#)**  
Use the Extract policy to extract and transform data from fields in the API context.
- **[GatewayScript](#)**  
Use the gatewayscript policy to execute a specified DataPower GatewayScript program.
- **[Generate JWT](#)**  
Use the Generate JWT security policy in IBM API Connect to generate a JSON Web Token (JWT).
- **[Validate JWT](#)**  
Use the Validate JWT security policy to enable the validation of a JSON Web Token (JWT) in a request before allowing access to the APIs.
- **[GraphQL introspect](#)**  
Use the GraphQL introspect policy to introspect a GraphQL schema.
- **[Invoke](#)**  
Apply the Invoke policy to call another service from within your assembly. The response from the backend is stored either in the variable `message.body` or in the response object variable if it is defined. The policy can be used with JSON or XML data, and can be applied multiple times within your assembly.
- **[JSON to XML](#)**  
Use the JSON to XML policy to convert the context payload of your API from the JavaScript Object Notation (JSON) format to the extensible markup language (XML) format.
- **[Log](#)**  
Use the Log policy to customize or override the default activity logging configuration for an API.
- **[Map](#)**  
Use the Map policy to apply transformations to your assembly flow and specify relationships between variables.
- **[OAuth](#)**  
An OAuth policy performs the requested OAuth processing based on the defined OAuth provider settings.
- **[Parse](#)**  
Use the Parse policy to control the parsing of an input document. When the input document is a JSON string, the string is parsed instead of copied over.
- **[Proxy](#)**  
Apply the Proxy policy to invoke another API within your assembly, particularly if the separate API contains a large payload. The response from the backend is stored in the `message.body` and in the response object variable if it is defined. Only one policy is permitted to be run per unique assembly flow.
- **[Rate Limit](#)**  
Use the Rate Limit policy to apply one or more rate, burst, or count limits at any point in your API assembly flow. Rate and burst limits restrict the number of calls made to an API in a specified time period, while count limits impose a strict limit on the total number of calls.
- **[Redaction - DataPower API Gateway](#)**  
Use the Redaction policy to completely remove or to redact specified fields from the Request body, the Response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.
- **[Redaction - DataPower Gateway \(v5 compatible\)](#)**  
Use the Redaction policy to completely remove or to redact specified fields from the Request body, the Response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.
- **[Set Variable](#)**  
Use the Set Variable policy to set the value of a runtime variable, or to clear a runtime variable, or to add a header variable.
- **[User Security](#)**  
Use the User Security policy to extract a user's credentials, authenticate those credentials, and obtain authorization from the user.
- **[Validate - DataPower API Gateway](#)**  
Use the Validate policy to validate the payload in an assembly flow against a schema.
- **[Validate - DataPower Gateway \(v5 compatible\)](#)**  
Use the Validate policy to validate the payload in an assembly flow against a JSON or an XML schema.
- **[Validate Username Token](#)**  
Use the Validate Username Token policy to validate a Web Services Security (WS-Security) UsernameToken in a SOAP payload before allowing access to the protected resource.

- [Websocket Upgrade](#)  
Use the Websocket Upgrade policy to process API requests and responses through a WebSocket connection.
- [XML to JSON](#)  
Use the XML to JSON policy to convert the context payload of your API from the extensible markup language (XML) format to JavaScript Object Notation (JSON).
- [XSLT](#)  
Use the XSLT policy to apply an XSLT transform to the payload of the API definition.

## Activity Log

Use the Activity Log policy to configure your logging preferences for the API activity that is stored in IBM® API Connect analytics. The preferences that you specify will override the default settings for collecting and storing details of the API activity.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [activity-log](#).

Note that if you are using the DataPower API Gateway, you can configure your logging preferences by using the API design editor. For more information, see [Configuring activity logging](#) (OpenAPI 2.0) or [Configuring activity logging](#) (OpenAPI 3.0).

## About

An API event record exists for each API execution event in the Gateway server. By default, the content type that is collected and stored in API event records is **activity** for when API execution completes successfully, and **payload** for when API execution completes with an error code. Apply the Activity Log policy to your assembly to change the type of content to log in these API event records. For more information about API event records, see [API event record fields](#).

Note: If payload logging is enabled, for the gateway to capture payloads buffering must also be enabled. Enable buffering as follows in the API YAML:

```
x-ibm-configuration:
 ...
 activity-log:
 enabled: true
 error-content: payload
 success-content: payload
 ...
 buffering: true
```

Note:

Activity log policies that call for logging of Analytics data upon success do not apply for the OAuth provider. The OAuth provider logs Analytics data for failure cases, but does not log successful cases.

## Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Activity Log policy properties

Property label	Required	Description	Data type
Title	Yes	A title for the policy is required, but a default value, <b>activity-log</b> is provided.	string
Description	No	A description of the policy.	string
Content	Yes	Defines the type of content to be logged when the operation is successful. Valid values: <ul style="list-style-type: none"> <li>• <b>none</b>: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal.</li> <li>• <b>activity</b>: Logs invocation only (only the resource URI is recorded).</li> <li>• <b>header</b>: Logs activity and header.</li> <li>• <b>payload</b>: Logs activity, header, and payload (the original request, if any, and the final response).</li> </ul> The default value is <b>activity</b> .	string
Error content	No	Indicates what content to log if an error occurs. Valid values: <ul style="list-style-type: none"> <li>• <b>none</b>: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal.</li> <li>• <b>activity</b>: Logs invocation only (only the resource URI is recorded).</li> <li>• <b>header</b>: Logs activity and header.</li> <li>• <b>payload</b>: Logs activity, header, and payload (the original request, if any, and the final response).</li> </ul> The default value is <b>payload</b> .	string

## Related tasks

- [Creating a new REST OpenAPI definition](#)

## Related information

- [API Analytics](#)

## Client Security

The Client Security policy can be used to extend client authentication access for your APIs.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [client-security](#).

## About

You use the Client Security policy to create an assembly action that's separate from the OpenAPI specification, to allow for more options to authenticate an application. Note that you cannot use the client-security policy to perform rate-limiting enforcement.

In a client security action, you define the following settings.

- Control whether to stop processing if client security fails. When client security fails, stops assembly processing and returns an error.
- Control whether to require the client secret. When required, the secret is compared to the registered secret on the application that is identified by the client ID.
- Define the method to extract client credentials from the request.
  - For all methods except HTTP, use the ID Name and the Secret Name properties to specify the location that contains the ID and the location that contains the secret.
    - When Cookie, specify which cookie.
    - When Context Variable, specify which runtime context variable.
    - When Form, specify the form data.
    - When Header, specify which header.
    - When Query, specify which query parameter.
  - For the HTTP method, use the HTTP Type property to specify the format of the authorization header, which expects the Basic form in the **basic base64\_id:secret** format.
- Define the method to authenticate the extracted client credentials. The supported methods are Native (this means by using IBM® API Connect), or by using a specified Third party user registry.
  - If Third party, use the User Registry Name property to specify the user-registry to authenticate the extracted client credentials. The supported registry types are LDAP and authentication URL.

## Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Client Security policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is <b>client-security</b> .	string
Description	No	A description of the policy.	string
Stop on Error	Yes	If this setting is enabled, assembly processing stops if client security fails, and an error is returned. The check box is selected by default.	boolean
Secret Required	Yes	If this setting is enabled, the client secret must be sent in the request. The secret is compared to the registered secret on the application that is identified by the client ID. The check box is selected by default.	boolean
Credential Extraction Method	Yes	Select one of the following options to define how the calling application authenticates: <ul style="list-style-type: none"><li>• Header: client ID and client secret credentials must be supplied in the request header.</li><li>• Query: client ID and client secret credentials must be supplied as query parameters in the request URL.</li><li>• Form: client ID and client secret credentials must be supplied as form data sent in a POST request.</li><li>• Cookie: client ID and client secret credentials must be supplied in a header called <b>Cookie</b>.</li><li>• HTTP: the calling application must authenticate by using basic authentication.</li><li>• Context Variable: the credentials that are used for authenticating the client are obtained from context variables that you set in the assembly flow prior to the Client Security policy, by using a <a href="#">GatewayScript</a> policy for example. The names of these context variables are determined by the values that you supply in the ID Name and Secret Name properties of the Client Security policy.</li></ul> The default value is Header.	string

Property label	Required	Description	Data type
ID Name	Yes, unless the selected value of Credential Extraction Method is HTTP	The name of the parameter whose value specifies the client ID. For all Credential Extraction Method options other than Context Variable and HTTP, the calling application must supply a parameter with this name, in the location defined by the Credential Extraction Method option. For the Context Variable option, this property specifies the name of a context variable. This property does not apply if the Credential Extraction Method is HTTP.	string
Secret Name	Yes, if Secret Required is enabled and the selected value of Credential Extraction Method is other than HTTP	The name of the parameter whose value specifies the client secret. For all Credential Extraction Method options other than Context Variable and HTTP, the calling application must supply a parameter with this name, in the location defined by the Credential Extraction Method option. For the Context Variable option, this property specifies the name of a context variable. This property does not apply if the Credential Extraction Method is HTTP.	string
HTTP Type	Yes, if the selected value of Credential Extraction Method is HTTP	The authentication type. Currently, the only available option is Basic.	
Authenticate Client Method	Yes	Select one of the following options: <ul style="list-style-type: none"> <li>Native: only client ID and client secret are used to authenticate the client. If the selected value of Credential Extraction Method is HTTP then the calling application must supply the client ID for the user name, and the client secret for the password.</li> <li>Third party: a user registry is used to authenticate the client. If the selected value of Credential Extraction Method is other than HTTP then the calling application must supply the user name for the client ID, and the password for the client secret.</li> </ul> <p>The default value is Native.</p>	string
User Registry Name	Yes, if the selected value of Authenticate Client Method is Third party	Select the user registry that will be used to authenticate the client. The supported registry types are LDAP and authentication URL.	string

API Gateway only

## Extract

Use the Extract policy to extract and transform data from fields in the API context.

## Before you begin

Table 1. Gateways that support this policy and the corresponding policy versions

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in the assembly editor. For details about how to configure the policy in your OpenAPI source, see [extract](#).

## About this task

The Extract policy specifies the data source that contains the content to transform, the fields that contain the content, and expressions that define how to transform the content. You use a subset of JSONata notation to specify the fields to extract and transform. For more information, see [Constructing JSONata expressions to extract and transform data](#).

The input to the Extract policy must be parsed data. One way to produce parsed data is to add a Parse policy before the Extract policy in your assembly.

## Procedure



- In the navigation pane, click  Develop, then select the APIs tab. The Develop page opens.
- Click the title of the API definition that you want to work with.
- Select the Gateway tab, then click Policies in the navigation pane. For more information about working with the assembly editor for an API, see [The assembly editor](#).
- Find the Extract policy in the palette, and drag the policy onto your canvas.
- Specify the following properties.

Table 2. Extract policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is <b>extract</b> .	string
Description	No	A description of the policy.	string
Root	Yes	The data source that contains the content to transform. The default value is <b>message.body</b> .	string
Capture	Yes	The path expression that identifies the field. The default setting is <b>\$</b> , which indicates the entire input.	string
Transform	Yes	The expression that defines how to transform the content.	string

- Specify a version for the policy by clicking the Source icon , and completing the **version** section of the policy YAML. For example:

```

execute:
- extract:
 version: 2.0.0
 title: extract
...

```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

## Example

Transform the contents of the `account` field to include only the last 4 characters.

```

- extract:
 version: 2.0.0
 title: extract
 root: message.body
 extracts:
 - capture: $.members.policy.**.account
 transform: $substring($,-4)
 description: Include only the last 4 characters of the account field.

```

- [Constructing JSONata expressions to extract and transform data](#)

To extract and transform data from fields in the API context when using the Extract policy with the DataPower API Gateway, you supply a JSONata expression that defines the fields to extract and transform.

API Gateway only

## Constructing JSONata expressions to extract and transform data

To extract and transform data from fields in the API context when using the Extract policy with the DataPower® API Gateway, you supply a JSONata expression that defines the fields to extract and transform.

Note: For a list of known limitations to DataPower API Gateway support for JSONata, see [Limitations to support for JSONata](#). The following JSONata functions are supported:

- Aggregation functions:
  - `$average(array)`
  - `$max(array)`
  - `$min(array)`
  - `$sum(array)`
- Array functions:
  - `$append(array1, array2)`
  - `$count(array)`
  - `$reverse(array)`
  - `$sort(array [, function])`
  - `$zip(array1, ...)`
- Boolean functions:
  - `$boolean(arg)`
  - `$exists(arg)`
  - `$not(arg)`
- Numeric functions:
  - `$abs(number)`
  - `$ceil(number)`
  - `$floor(number)`
  - `$formatBase(number [, radix])`
  - `$number(arg)`
  - `$power(base, exponent)`
  - `$round(number [, precision])`
  - `$sqrt(number)`
- Object functions:
  - `$keys(object)`
  - `$lookup(object, key)`
  - `$merge(array<object>)`
  - `$spread(object)`
  - `$type(value)`

In addition to the standard values returned by `$type()`, the API gateway implementation of `$type()` can return the following values: `binary`, `empty`, `graphql`, `stream`, or `xml`.
- String functions:
  - `$contains(str, pattern)`
  - `$join(array[, separator])`
  - `$length(str)`
  - `$lowercase(str)`
  - `$match(str, pattern [, limit])`
  - `$pad(str, width [, char])`
  - `$replace(str, pattern, replacement [, limit])`
  - `$split(str, separator [, limit])`
  - `$string(arg)`

- `$substring(str, start[, length])`
- `$substringAfter(str, chars)`
- `$substringBefore(str, chars)`
- `$trim(str)`
- `$uppercase(str)`

- You can use the following functional extensions to standard JSONata notation. Each extension corresponds to a part of the API context.

Table 1. Functional extensions to JSONata

Extension	Variable	Description
<code>\$header (name )</code>	<code>message.headers.name</code>	Message header
<code>\$httpVerb ()</code>	<code>request.verb</code>	HTTP method of the request
<code>\$operationID ()</code>	<code>api.operation.id</code>	ID of the operation
<code>\$operationPath ()</code>	<code>api.operation.path</code>	Path of the operation
<code>\$queryParameter ('name')</code>	<ul style="list-style-type: none"> <li>◦ <code>request.parameters.name.locations</code> The supported keyword is <code>query</code>.</li> <li>◦ <code>request.parameters.name.values</code></li> </ul>	Searches for the index of <code>query</code> in <code>request.parameters.name.locations</code> and returns <code>request.parameters.name.values[index]</code> , where <code>[index]</code> is the value for <code>query</code> in locations. Parameter values are not URL decoded.
<code>\$statusCode ()</code>	<code>message.status.code</code>	Status code
<code>\$storageType ([arg])</code>	<code>variable.body</code> You can specify any variable in the API context. When no variable is specified, the default variable <code>message.body</code> is used.	Storage type of the message. The supported values are <code>binary</code> , <code>empty</code> , <code>graphql</code> , <code>json</code> , <code>stream</code> , or <code>xml</code> .
<code>\$urlParameter ('name')</code>	<ul style="list-style-type: none"> <li>◦ <code>request.parameters.name.locations</code> The supported keywords are <code>path</code> and <code>query</code></li> <li>◦ <code>request.parameters.name.values</code></li> </ul>	Searches for the index of <code>path</code> and <code>query</code> in <code>request.parameters.name.locations</code> and returns a single array that contains both <code>path</code> and <code>query</code> values from <code>request.parameters.name.values</code> . When the URL contains both path and query parameter values, the array includes the path values first followed by the query values. The values of each parameter type are added in the order that they are received. Parameter values are URL decoded. For example, the following URL contains both path and query parameter values. <code>http://example.com/petstore/cats/adopt?breed=Sphynx&amp;breed=Siamese</code> The <code>\$urlParameter ('breed')</code> URL returns the following array of values. <code>[cats, adopt, Sphynx, Siamese]</code> In this example, the URL includes an API path that is configured as <code>/petstore/{breed}/{breed}</code> , where <code>breed</code> is configured to be a path parameter of the API path. As a result, <code>cats</code> and <code>adopt</code> are included in the output.
<code>\$xpath (path, xpathExpression)</code>	You can specify any writable variable in the API context. The <code>xpathExpression</code> must be a literal string.	Allows use of XPath expressions. The following example specifies all price elements in the source. <code>\$xpath (\$, '//price')</code>

Table 2. Functional extensions to JSONata for GraphQL

Extension	Variable	Description
<code>\$gqlActiveOperation ([graphql_message])</code>	<code>message.body</code>	Gets the active operation found in the specified GraphQL message. The <code>operationName</code> must be the same as the name of the active operation.
<code>\$gqlAlias (graphql_field_node)</code>	<code>message.body</code>	Gets the alias of a GraphQL field node.
<code>\$gqlFragments ([graphql_message])</code>	<code>message.body</code>	Gets the fragments found in the specified GraphQL message.
<code>\$gqlName ([graphql_node])</code>	<code>message.body</code>	Gets the node name. For operations, the node name is the <code>operationName</code> . For fields, fragment definitions, arguments, and other elements, the node name is the name of the element. By default, the <code>operationName</code> of <code>message.body</code> is retrieved.
<code>\$gqlOperations ([graphql_message])</code>	<code>message.body</code>	Gets the operations found in the specified GraphQL message.
<code>\$gqlType ([graphql_node])</code>	<code>message.body</code>	The operation type of the active operation is retrieved for Query, Mutation, and Subscription query types.

- You can use the `&` (concatenation) navigation operator.

You can use the following JSONata numeric operators:

- `+` (addition)
- `-` (subtraction)
- `*` (multiplication)
- `/` (division)
- `%` (modulo)

You can use the following JSONata comparison operators for number values or strings:

- `=`
- `!=`
- `<`

- >
- <=
- >=

You can also use the following operators and expressions.

- Parentheses to convert a sequence into an array, specify operator precedence, or compute complex expressions on a context value.
- Array ranges and predicate expressions.
- Single asterisk (\*) and double asterisk (\*\*) wildcard characters.

The following elements of a GraphQL query can be exposed in JSONata notation using the syntax shown.

- **query**  
The entire GraphQL query including operations and fragments.
- **operationName**  
For an anonymous operation, **operationName** can be empty.
- **~fragmentSpreadName**
- **on~typeCondition**
- **~~fragmentDefinitionName**

## GatewayScript

Use the gatewayscript policy to execute a specified DataPower® GatewayScript program.

### Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [gatewayscript](#).

### About

The gatewayscript policy gives you built-in access to the DataPower Gateway module via variable **apim**.

### Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. gatewayscript policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is <b>gatewayscript</b> .	string
Description	No	A description of the policy.	string
Source	Yes	The GatewayScript source code to execute. For example: <pre>var message = [ 'Hello', 'World!' ]; console.debug(message.join(' '));</pre>	string

### Examples

The following examples show how the full OpenAPI for the policy looks in the source code.

Example one:

```
gatewayscript:
 title: writes message to DataPower log
 source: console.debug('Hello World!');
```

Example two:

```
gatewayscript:
 title: script written in multiple lines
 source: |
 var message = ['Hello', 'World!'];
 console.debug(message.join(' '));
```

For more code examples, see [GatewayScript code examples](#) and, if you are using the DataPower API Gateway, [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway](#).

For general information on using GatewayScript, see the following topics in the DataPower product documentation:



- [GatewayScript APIs for API management](#)
- [Accessing and manipulating a variable in the API context](#)
- [OAuth context variables](#)

## Errors

The following error can be thrown while the policy is being executed:

- **JavaScriptError** - a generic error that captures all errors that occur during the execution of the policy.
- [GatewayScript code examples](#)  
Example GatewayScript code snippets to help the creation of a gatewayscript policy.

## Related tasks

- [Creating a new REST OpenAPI definition](#)

## GatewayScript code examples

Example GatewayScript code snippets to help the creation of a gatewayscript policy.

Attention:

- The mechanisms described here for interacting with the API Connect context are intended for v5-compatible gateways and also to provide v5 compatibility for APIs created for the API Gateway. However, if you are writing a GatewayScript policy or user-defined policy for a new API Gateway API, use the mechanisms described in [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway](#) for better performance and long-term support. For API Gateway APIs, the v5-compatibility functions listed here are to be used for v5 migration only.
- The GatewayScript functions that are listed here use the common name **apim** to call the methods. However, you can change the common name to one of your choice by using one or other of the following function calls depending on your gateway type:

```

DataPower Gateway (v5 compatible) var name = require('local:///isp/policy/apim.custom.js'); // v5-Compatible Gateway
API Gateway var name = require('apim'); // DataPower API Gateway

```

where *name* is your chosen name; for example:

```
var apim = require('apim');
```

## Access the assembly context

The following code snippets show examples of how to access the assembly context.

Example one returns the **request** context:

```
apim.getvariable('request');
```

Example two returns the header named **myheader**:

```
apim.getvariable('request.headers.myheader');
```

Example three shows how to set, add, or clear a context variable, in this case a message header:

```
apim.setvariable('message.headers.name', value, action)
```

where

- *name* is the name of the message header that you want to set, add, or clear.
- *value* is the string value that you want to set the variable to. This can be a literal value, or another variable. For example, to set your named variable to the value of the Content-Type header in a request, you would specify the *value* as `request.headers.content-type`. This property is only required when **set** or **add** is specified as the action.
- *action* is the action that you want to apply to the variable. Valid options are:
  - **set**
  - **add**
  - **clear**

If no option is set, the default option of **set** is applied.

For a complete list of context variables, see [API Gateway context variables](#), and for more information about how to reference context variables in IBM API Connect see [Variable references in API Connect](#). If you are using the DataPower® API Gateway, see also [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway](#).

## Read input synchronously

The following code snippets show examples of how to read input synchronously by using the **apim.getvariable** function to read data direct from a context. JSON input is returned as a JavaScript object. XML data is returned as a NodeList object (DOM).

Example one returns a synchronous read of the original request body into the variable that is called *json*:

```
var json = apim.getvariable('request.body');
```

Example two returns a synchronous read of the current message into the variable that is called *xml*:

```
var xml = apim.getvariable('message.body');
```

Note: If the content type of the data is neither XML nor JSON, the `apim.getvariable()` function returns a `NodeList` object consisting of one binary node, which must be converted to a string. If the content really is XML or JSON, you must then parse this string, as illustrated in the following example:

```
var data = apim.getvariable('request.body');

// if a nodelist was returned and the type is 13 (BLOB/Binary Node), convert it
// to a Buffer, which can then be converted to a string
if ((data.item && data.length > 0 && data.item(0).nodeType === 13)) {
 data = data.item(0).toBuffer().toString();
}

// if the string really is XML or JSON, then now add code to parse it with
// the appropriate XML or JSON parse function
.
.
.
```

## Read input asynchronously

The following code snippets show the `apim.readInput()` calls that you can use to perform a JavaScript callback function to read input data asynchronously into a variable. JSON input is returned as a JavaScript object. XML data is returned as a `NodeList` object (DOM). Other types of input are returned as a `Buffer` object.

```
apim.readInput(callback(err,input) {});
apim.readInputAsJSON(callback(err,json) {});
apim.readInputAsXML(callback(err,nodelist) {});
apim.readInputAsBuffer(callback(err,buffer) {});
```

Note: The `apim.readInput()` function attempts to determine the input data type and call the appropriate `apim.readInputAsType()` function in order to return the data in the correct format. However, to ensure that the correct data type is returned, use the `apim.readInputAsType()` function.

The `apim.readInputAsType()` function reads the data from either the `INPUT` context (for example `session.INPUT`), or from a policy output context (for example `policy.output`), depending on whether a previous policy had been called that had written output to a policy output context. This function then calls the underlying IBM DataPower GatewayScript `readAsType` method on the relevant context to read the data, and then use JavaScript callbacks to return the data to a variable. For more information about DataPower methods, see [Gateway programming model and GatewayScript](#).

The following is an example of how to use the `apim.readInputAsJSON()` callback function to get data into a variable that is called `json`:

```
apim.readInputAsJSON(function (error, json) {
 if (error)
 {
 // handle error
 }
 else
 {
 // the json parameter will contain the json data that has been read
 // process the json object in some way and write to the output context
 if (json.test==='true')
 {
 // if json data contains a variable called test that is set to true, then also add some test data
 json.data = 'This is my test data'
 }
 session.output.write(json);
 // write to the output context
 }
});
```

## Configure error information

The following code block shows an example of how to configure the policy implementation to produce error information by using the `apim.error()` function. In this example, `MyError` is thrown to the assembly flow. If there is a catch handler it catches this error, otherwise the assembly skips the execution of the flow and sends a `500 Internal Error` response.

```
apim.error('MyError', 500, 'Internal Error', 'Some error message');
```

where:

- `MyError` is the name of the error.
- `500` is the HTTP code of the required error message.
- `Internal Error` is the HTTP reason phrase for the error.
- `Some error message` is the suggested action for the user.

## Accessing the caught exception in a catch block

The following example shows how, in the `catch` block of an API assembly, you can obtain the details of the current caught exception. A possible use would be to create a custom error response using the details of the caught exception.

```
let exception = apim.getError();
```

The function returns a JSON object; for example:

```
{
 "name": "OperationError",
 "message": "This is a thrown Operation Error",
 "policyTitle": "Throw Operation Error",
 "status": {
 "code": "500",
 "reason": "Internal Server Error"
 }
}
```

## Write output

The following example shows how to write data into the output, in this case the JSON data '{ "status": "created" }', by using the `session.output.write` variable:

```
session.output.write('{ "status": "created" }');
```

The `session.output.write` variable is a standard GatewayScript method that writes data to the output context. For more information, see [Contexts and sessions](#). Use the following function to set the format of the content that is written to the `session.output` variable:

```
apim.output('application/type');
```

For example, if you were creating a policy that called:

```
session.output.write('<test>this is some xml data</test>');
```

the policy would write XML data to the output context. You would need to configure the policy to also call

```
apim.output('application/xml');
```

to tell the system that the output is XML. It can cause issues in the processing of the policy if the actual output context format, and the format that is specified in `apim.output`, are different.

Note: If you use `apim.setvariable` to manipulate `message.body`, you must immediately follow the `apim.setvariable` function call with an `apim.output` function call that specifies the content type of the payload that was written to `message.body` by `apim.setvariable`.

## Related information

- 🔗 [GatewayScript APIs for API management](#)
- 🔗 [Accessing and manipulating a variable in the API context](#)
- 🔗 [OAuth context variables](#)

## Generate JWT

Use the Generate JWT security policy in IBM® API Connect to generate a JSON Web Token (JWT).

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [jwt-generate](#).

## About

JSON Web Token (JWT) is a compact, URL-safe way of representing claims that are to be transferred between two parties. The Generate JWT policy enables you to generate claims and configure whether they are to be used as the payload of a JSON Web Signature (JWS) structure, or as the plain text of a JSON Web Encryption (JWE) structure. Specifying the cryptographic material for both the JWS and the JWE produces a nested JWT that is both digitally signed and encrypted. The JWT is then assigned to the Authorization header as a Bearer token (the default option), or to the runtime variable in the JSON Web Token (JWT) property, if specified.

Note:

- For algorithm types HS256, HS384, and HS512 the cryptographic objects referenced must be a Shared Secret Key.
- For algorithm types RS256, RS384, RS512, ES256, ES384, ES512, PS256, PS384, PS512 the cryptographic objects referenced must be a Crypto Key (private key).
- The cryptographic material can be provided through a JSON Web Key (JWK).
- If both a cryptographic object and a JWK are specified, the cryptographic object is used to sign the JWT.

## Prerequisites

The following prerequisites apply:

- If you are using one or more cryptographic objects, they must be located in the API Connect domain on the DataPower appliance. The cryptographic objects must reference the Shared Secret Key or certificate that is needed to encrypt or sign the JWT contents.
- If a JSON Web Key (JWK) is being used, it must be referenced by a runtime variable.

## Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Generate JWT policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is <code>jwt-generate</code> .	string

Property label	Required	Description	Data type
Description	No	A description of the policy.	string
JSON Web Token (JWT)	No	Runtime variable in which to place the JWT that is generated. The default value is: <b>generated.jwt</b> . However, if not set, the JWT that is generated is written to the Authorization Header as a Bearer token.	string
JWT ID Claim	No	Indicates whether a JWT ID (jti) claim should be added to the JWT. If selected, the property is set to <b>true</b> , and a UUID is generated and set as the JTI claim value.	boolean
Issuer Claim	Yes	Runtime variable from which the Issuer (iss) claim string can be retrieved. This claim represents the Principal that issued the JWT. The default value is: <b>iss.claim</b>	string
Subject Claim	No	Runtime variable from which the Subject (sub) claim string can be retrieved.	string
Audience Claim	No	Runtime variable from which the Audience (aud) claim string can be retrieved. Multiple variables are set by using a comma-separated string.	string
Validity Period	Yes	The length of time (in seconds), that is added to the current date and time, in which the JWT is considered valid. The default value is <b>3600</b> .	integer
Private Claims	No	Runtime variable from which a valid set of JSON claims can be retrieved. These claims are added to any set of claims specified previously.	string
Sign JWK variable name	No	Runtime variable that contains the JWK that is used to sign the JWT. <sup>1</sup>	string
Cryptographic Algorithm	No	The cryptographic algorithm to use. Valid values are: <ul style="list-style-type: none"> <li>• HS256</li> <li>• HS384</li> <li>• HS512</li> <li>• RS256</li> <li>• RS384</li> <li>• RS512</li> <li>• ES256</li> <li>• ES384</li> <li>• ES512</li> <li>• PS256</li> <li>• PS384</li> <li>• PS512</li> </ul>	string
Sign Crypto Object	No	The cryptographic object to use to sign the JWT. <sup>1</sup>	string
Encryption Algorithm	No	The encryption algorithm to use. Valid values are: <ul style="list-style-type: none"> <li>• A128CBC-HS256</li> <li>• A192CBC-HS384</li> <li>• A256CBC-HS512</li> </ul>	string
Encrypt JWK variable name	No	Runtime variable that contains the JWK to use to encrypt the JWT.	string
Key Encryption Algorithm	No	The key encryption algorithm to use. Valid values are: <ul style="list-style-type: none"> <li>• RSA1_5</li> <li>• RSA-OAEP</li> <li>• RSA-OAEP-256</li> <li>• dir</li> <li>• A128KW</li> <li>• A192KW</li> <li>• A256KW</li> </ul>	string
Encrypt Crypto Object	No	The cryptographic object to use to encrypt the claim.	string

## Example

```

- jwt-generate:
 version: 1.0.0
 title: jwt-generate
 iss-claim: iss.claim
 exp-claim: 3600
 jwt: generated.jwt
 jti-claim: true
 sub-claim: sub.claim
 aud-claim: aud.claim
 private-claims: private.claims
 jws-jwk: jws.jwk
 jws-alg: HS256
 jws-crypto: jwsCryptoObjectName
 jwe-enc: A128CBC-HS256
 jwe-jwk: jwe.jwk
 jwe-alg: A128KW
 jwe-crypto: jweCryptoObjectName

```

## Errors

The following error can be thrown while the policy is being executed:

- **RuntimeError** - a generic error that captures all errors that occur during the execution of the policy. Upon failure, the detailed error message that is received from the underlying JOSE module is written to the default system log as an error message. This detailed error message is also assigned to the runtime variable `jwt-generate.error-message`, so it can be retrieved via `catch`.

If a `catch` is not configured, in the case of a failure the Generate JWT policy returns an HTTP code 500 **Invalid-JWT-Generate** failure. The detailed error message from the underlying JOSE module can be found in the system log.

Attention: If you are an API developer who is troubleshooting a failure that one of your customers had with your API, consider the security risks before sending a customer the exact content of the log message. You can avoid the possibility of someone launching an attack based on the information that they receive from the log message by sending the customer only general information about the message.

## Related information

- [Introduction to JSON Web Tokens](#)

<sup>1</sup> A JWK and a Crypto Object are both valid ways of providing the cryptographic data necessary to sign the JWT. However, if both data types are specified, only the Crypto Object is used.

## Validate JWT

Use the Validate JWT security policy to enable the validation of a JSON Web Token (JWT) in a request before allowing access to the APIs.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [jwt-validate](#).

## About

JSON Web Token (JWT) is a compact, URL-safe way of representing claims that are to be transferred between two parties. The Validate JWT policy enables you to secure access to your APIs by using JWT validation. For example, when an input request that contains a JWT in the header is received, the Validate JWT policy extracts the token, verifies, and decrypts (if appropriate) the signature, and validates the claim. If valid, the claim is put in a runtime variable (for subsequent use if required), and access is allowed to the API. If the claim is not valid, access is denied.

All claims that are specified in the Validate JWT policy are validated, but this is not necessarily all the claims that are contained in the JWT. Not all claims that are in the JWT must be validated, but if any one of the claims that are specified in the Validate JWT policy fail, the whole validation fails. If the validation succeeds, the full set of claims that are contained in the JWT are written to the runtime variable specified in the Output Claims property. Thereby allowing any subsequent action to use this runtime variable to further validate the full set of claims that were in the JWT, as necessary.

Note:

- If the original message was signed with a Shared Secret Key, the cryptographic object that is specified must also be a Shared Secret Key.
- If the original message was signed with a Private Key, the cryptographic object that is specified must be a Crypto Certificate (public certificate).
- The cryptographic material can be provided through a JSON Web Key (JWK).
- If a JWK header parameter is included in the header of the JWT, the parameter must match the JWK or cryptographic object that is specified in the policy, or the JWT validation will fail.
- If both a cryptographic object and a JWK are specified, the cryptographic object is used to decrypt or verify the JWT.
- The JWT validate action on the DataPower API Gateway can verify a JWT by using either a single JWK, or a JWK set.

## Prerequisites

The following prerequisites apply:

- IBM® DataPower V7.5 with the Application Optimization (AO) option.
- If you are using one or more cryptographic objects, they must be located in the IBM API Connect domain on the DataPower appliance. The cryptographic objects must reference the Shared Secret Key or public certificate that is needed to decrypt the JWT contents or verify the signature.
- If a JSON Web Key (JWK), or JWK set, is being used, it must be referenced by a runtime variable.

## Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Validate JWT policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is <code>jwt-validate</code> .	string
Description	No	A description of the policy.	string

Property label	Required	Description	Data type
JSON Web Token (JWT)	Yes	Context or runtime variable that contains the JWT to be validated. The default value is: <code>request.headers.authorization</code> . However, if this property is not set, the policy looks for the JWT in the <code>request.headers.authorization</code> location by default.  Note: The format of the authorization header must be:  <code>"Authorization: Bearer jwt-token"</code>  where <code>jwt-token</code> is the encoded JWT.	string
Output Claims	Yes	Runtime variable to which the full set of claims that are contained in the JWT is assigned. The default value is: <code>decoded.claims</code> .	string
Issuer Claim	No	The Perl Compatible Regular Expressions (PCRE) to use to validate the Issuer (iss) claim.	string
Audience Claim	No	The PCRE to use to validate the Audience (aud) claim.	string
Decrypt Crypto Object	No	The cryptographic object (a shared key or certificate) to use to decode the claim. <sup>1</sup>	string
Decrypt Crypto JWK variable name	No	Runtime variable that contains the JWK to use to decrypt the JWT. <sup>1</sup>	string
Verify Crypto Object	No	The cryptographic object (a shared key or certificate) to use to verify the signature. <sup>2</sup>	string
Verify Crypto JWK variable name	No	Runtime variable that contains the JWK, or JWK set, to use to verify the signature. <sup>2</sup>	string

## Example

```
- jwt-validate:
 version: 1.0.0
 title: jwt-validate
 jwt: request.headers.authorization
 output-claims: decoded.claims
 iss-claim: "'^data.*'"
 aud-claim: "'^id.*'"
 jwe-crypto: jweCryptoObjectName
 jwe-jwk: jwe.jwk
 jws-crypto: jwsCryptoObjectName
 jws-jwk: jws.jwk
```

## Errors

The following error can be thrown while the policy is being executed:

- **RuntimeError** - a generic error that captures all errors that occur during the execution of the policy. Upon a validation failure, the detailed error message that is received from the underlying JOSE module is written to the default system log as an error message. This detailed error message is also assigned to the runtime variable `jwt-validate.error-message`, so it can be retrieved via `catch`.

If a `catch` is not configured, in the case of a validation failure the Validate JWT policy returns an HTTP code 500 **Invalid-JWT-Validate** failure. The detailed error message from the underlying JOSE module can be found in the system log.

Attention: If you are an API developer who is troubleshooting a failure that one of your customers had with your API, consider the security risks before sending a customer the exact content of the log message. You can avoid the possibility of someone launching an attack based on the information that they receive from the log message by sending the customer only general information about the message.

For examples, see [jwt-validate](#).

## Related information

- [Introduction to JSON Web Tokens](#)

<sup>1</sup> A JWK and a Crypto Object are both valid ways of providing the cryptographic data necessary to decrypt the JWT. However, if both data types are specified, only the Crypto Object is used.

<sup>2</sup> A JWK, or JWK set, and a Crypto Object are both valid ways of providing the cryptographic data necessary to verify the JWT. However, if both data types are specified, only the Crypto Object is used.

## GraphQL introspect

Use the GraphQL introspect policy to introspect a GraphQL schema.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [graphql-introspect](#).

## Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. GraphQL introspect policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is <code>graphql-introspect</code> .	string
Description	No	A description of the policy.	string
Input	No	A variable in the API context that contains the input to introspect. The content of the <code>body</code> field of the variable is the input to introspect. By default, the variable name is <code>message</code> .	string
Output	No	A variable in the API context where the results of the introspection are stored. The content of the <code>body</code> field of the variable is the result of GraphQL introspection. The default variable name is the same as that of the variable specified in the Input field. Therefore, by default, the input of the introspection is overwritten by the output.	string

## Invoke

Apply the Invoke policy to call another service from within your assembly. The response from the backend is stored either in the variable `message.body` or in the response object variable if it is defined. The policy can be used with JSON or XML data, and can be applied multiple times within your assembly.

Note: The Invoke policy does not support responses with multipart form data, that is, when the response is set to `Content-Type: multipart/related`.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.1.1 or later) 2.2.0 (DataPower API Gateway Version 10.0.2.0 or later) 2.3.0 (DataPower API Gateway Version 10.0.3.0 or later)

- [Configuring the Invoke policy for DataPower API Gateway](#)  
Follow these steps to configure the Invoke policy for DataPower API Gateway in the assembly user interface.
- [Configuring the Invoke policy for DataPower Gateway \(v5 compatible\)](#)  
Follow these steps to configure the Invoke policy for DataPower Gateway (v5 compatible) in the assembly user interface.

## Related tasks

- [Creating a new REST OpenAPI definition](#)
- [Handling errors in the assembly](#)

## Related information

- [TLS profiles](#)



## Configuring the Invoke policy for DataPower API Gateway

Follow these steps to configure the Invoke policy for DataPower® API Gateway in the assembly user interface.

## About this task

Note: This topic describes the Invoke policy implementation in the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), see [Configuring the Invoke policy for DataPower Gateway \(v5 compatible\)](#). For information about the different types of gateway, see [API Connect gateway types](#). For details on how to configure the policy in your OpenAPI source, see [invoke](#).

## Procedure

1. In the navigation pane, click Develop, then select the APIs tab.  
The Develop page opens.
2. Click the title of the API definition that you want to work with.
3. Select the Gateway tab, then click Policies in the navigation pane.  
For more information about working with the assembly editor for an API, see [The assembly editor](#).
4. Find the Invoke policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.


Table 1. Invoke policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is <b>invoke</b> .	string
Description	No	A description of the policy.	string
URL	Yes	Specifies a URL for the target service. For a SOAP API, a URL is added by default. Where possible, the Invoke URL value is pre-supplied from information that is defined in the imported WSDL.	string
Backend type	No	Select one of the following options to determine how the payload is sent to the backend.  Detect Detect the message type and send the payload to the backend accordingly. XML Package the payload as XML and send. If the message type is not XML, the operation fails. JSON Package the payload as JSON and send. If the message type is not JSON, the operation fails. Binary Package the payload as binary and send, regardless of the message type. GraphQL Package the payload as GraphQL and send. If the message type is not GraphQL, the operation fails.  The default value is Detect.	string
GraphQL send type(policy version 2.2.0 and later)	Yes, if Backend type is set to GraphQL or Detect, and HTTP Method is set to POST or Keep	Select one of the following values to determine how the GraphQL payload is sent to the back end.  Detect Detect the message type and send the payload to the back end accordingly. GraphQL Package the payload as GraphQL and send. If the message type is not GraphQL, the operation fails. JSON Package the payload as JSON and send. If the message type is not GraphQL, the operation fails.  Note: GraphQL send type is supported only if Backend type is set to GraphQL or Detect, and HTTP Method is set to POST or Keep.	string
TLS profile	No	Specifies a TLS profile to use for the secure transmission of data.	string
Timeout	Yes	The time to wait before a reply from the endpoint (in seconds). The default value is <b>60</b> .	integer
Follow redirects	No	Specifies the behavior if the back-end server returns the HTTP status code <b>301 Moved Permanently</b> . If you select this checkbox, the <b>invoke</b> policy follows the URL redirection by making a further call to the URL specified in the <b>Location</b> header in the response. If you clear this checkbox, the <b>invoke</b> saves the <b>301</b> status code and the API call is considered to be complete. Note: The <b>follow-redirect</b> property is supported only by the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), the <b>invoke</b> always follows the URL redirection; the <b>proxy</b> policy (not supported by the DataPower API Gateway) saves the <b>301</b> status code and completes the API call without following the URL redirection.	boolean
Username	No	The username to use for HTTP Basic authentication.	string
Password	No	The password to use for HTTP Basic authentication.	string
HTTP Method	Yes	The HTTP method to use for the Invoke. The following values are valid. <ul style="list-style-type: none"> <li>• Keep</li> <li>• GET</li> <li>• POST</li> <li>• PUT</li> <li>• DELETE</li> <li>• PATCH</li> <li>• HEAD</li> <li>• OPTIONS</li> </ul> The default value is <b>GET</b> . However, if set to <b>Keep</b> , or the property is removed from the source, the HTTP method from the incoming request is used.	string
HTTP Version (policy version 2.1.0 and later)	Yes	The HTTP version that will be used when connecting to the backend server. Select one of the following values. <ul style="list-style-type: none"> <li>• HTTP/1.0</li> <li>• HTTP/1.1</li> <li>• HTTP/2</li> </ul> The default value is HTTP/1.1.	string
HTTP/2 Required (policy version 2.1.0 and later)	No	Select this checkbox to require that the server selects HTTP/2 during a TLS connection, otherwise the DataPower API Gateway rejects the connection and fails the request. For a non-TLS connection, a warning is logged stating that the requirement can't be enforced. This setting applies only if HTTP Version is set to HTTP/2.	boolean
Compression	No	Select this checkbox to enable Content-Encoding compression on upload. The checkbox is cleared by default.	boolean



Property label	Required	Description	Data type
Cache Type	No	<p>The cache type determines whether to cache documents, honoring or overriding the HTTP Cache Control directives received in the response from the target URL. This property takes effect only when a response is received, otherwise the policy always returns the non-expired response that was previously saved in cache. Valid values are:</p> <p><b>Protocol</b> The cache behavior is determined by the Cache-Control headers on the response, in accordance with RFC 7234. To optimize performance, if the gateway receives more than one request for a resource that is not in the cache but could be cached when the response from the target URL is received, the gateway sends only one request to the target URL; the remaining requests are not processed until the response from the first request has been received and the cache behavior has been determined from this response. If the response indicates that caching is possible, the gateway responds to all waiting requests with the cached resource. If the response indicates that caching is not possible, the gateway sends all waiting requests to the target URL.</p> <p>Use this option only if you expect that responses from the target URL can be cached, in which case it should improve performance and limit the demand on the target URL. If, however, the target URL never indicates that the gateway should cache its response, performance might be impaired when compared to the No Cache option.</p> <p><b>No Cache</b> Responses from the target URL are not cached on the gateway regardless of any caching headers returned. In this case, every request from the client is sent to the target URL. Use this option if you do not want to cache any of the backend responses on the gateway, or if it is unlikely that a response from the target URL will allow caching through the Cache-Control header settings.</p> <p><b>Time to Live</b> This option is similar to the Protocol option except it allows you to specify the amount of time that you want the successful response from the invoke or proxy to remain in the cache. Use this option only if you expect that responses from the target URL can be cached.</p> <p>The default value is Protocol.</p>	string
Time to Live	No	<p>Specifies the amount of time in seconds that the response stays in the cache. Applies only if the property Cache type is set to <b>Time to Live</b>. Enter a value in the range 5 - 31708800. The default value is <b>900</b>.</p>	integer
Cache key	No	<p>Specifies the unique identifier of the document cache entry. If omitted, the entire URL string is used as the key.</p>	string
Allow WebSocket Upgrade (policy version 2.1.0 and later)	No	<p>Select this checkbox to allow an HTTP or HTTPS connection to be upgraded to the WebSocket protocol if the request received by the corresponding HTTP or HTTPS handler on the DataPower API Gateway uses WebSocket (ws) or WebSocket Secure (wss).</p> <p>Note:</p> <ul style="list-style-type: none"> <li>This setting is available only for GraphQL APIs.</li> <li>The Allow WebSocket Upgrade setting is deprecated. To allow WebSocket upgrade requests that manage API processing data, define an assembly WebSocket upgrade policy. For more information, see <a href="#">WebSocket Upgrade</a>.</li> </ul>	boolean
Inject proxy headers	No	<p>If you select this checkbox, the <b>invoke</b> policy injects the <b>X-Forwarded-For</b>, <b>X-Forwarded-To</b>, <b>X-Forwarded-Host</b>, and <b>X-Forwarded-Proto</b> headers to the request that is sent to the target URL. The checkbox is cleared by default.</p>	boolean
Decode Request Params	No	<p>If you select this checkbox, any request parameters that are referenced by a variable definition on the target URL of the invoke policy are URL-decoded. The checkbox is cleared by default.</p>	boolean
QueryParam Encode	No	<p>If you select this checkbox, all "+" characters in the query parameter values of the target URL are encoded to %2F. The checkbox is cleared by default.</p>	boolean
Keep Payload	No	<p>If you select this checkbox, the invoke policy sends a payload on an <b>HTTP DELETE</b> method. The checkbox is cleared by default.</p>	boolean
Restrict to HTTP 1.0 (policy version 2.0.0 only)	No	<p>If you select this checkbox, HTTP transactions are restricted to version 1.0. The check box is cleared by default.</p>	boolean
Allow chunked uploads	No	<p>If you select this checkbox, chunked-encoded documents are sent to the server. When the HTTP 1.1 protocol is used, the body of the document can be delimited by either <b>Content-Length</b> or chunked encoding. While all servers can interpret <b>Content-Length</b>, many applications fail to understand chunked-encoded documents. For this reason, <b>Content-Length</b> is the standard method. The use of <b>Content-Length</b> interferes with the ability of the DataPower Gateway to fully stream. If you must stream full documents to the target server, enable this property.</p> <p>When enabled, the server must be RFC 2616 compatible. Unlike all other HTTP 1.1 features that can be negotiated down at run time, you must know beforehand that the target server is RFC 2616 compatible.</p> <p>The check box is selected by default.</p> <p>Note: Chunked encoding is not supported by the HTTP 1.0 protocol.</p>	boolean

Property label	Required	Description	Data type
Persistent Connection (policy version 2.3.0 and later)	No	Select this checkbox to enable HTTP persistent connections, allowing connection re-use. The checkbox is selected by default.	boolean
Header control	No	Specifies the headers in <code>message.headers</code> that you want to copy to the target URL. To prevent headers from being copied, complete the following steps. <ul style="list-style-type: none"> <li>a. Select Blocklist.</li> <li>b. Click Add blocklist.</li> <li>c. In the empty field that is displayed, enter the header name.</li> <li>d. To add further headers, repeat the previous steps.</li> </ul> To specify headers that you want to be copied, complete the following steps. <ul style="list-style-type: none"> <li>a. Select Allowlist.</li> <li>b. Click Add allowlist.</li> <li>c. In the empty field that is displayed, enter the header name.</li> <li>d. To add further headers, repeat the previous steps.</li> </ul> The values that you specify are in regular expression format. For example, to specify the Content-Type header, enter <code>^Content-Type\$</code>  By default, Blocklist is selected, with no blocklist entries, meaning that <b>all</b> headers are copied.	string
Parameter control	No	Specifies the parameters in the incoming request that you want to be copied to the target URL. To prevent parameters from being copied, complete the following steps. <ul style="list-style-type: none"> <li>a. Select Blocklist.</li> <li>b. Click Add blocklist.</li> <li>c. In the empty field that is displayed, enter the parameter name.</li> <li>d. To add further parameters, repeat the previous steps.</li> </ul> To specify parameters that you want to be copied, complete the following steps. <ul style="list-style-type: none"> <li>a. Select Allowlist.</li> <li>b. Click Add allowlist.</li> <li>c. In the empty field that is displayed, enter the parameter name.</li> <li>d. To add further parameters, repeat the previous steps.</li> </ul> The values that you specify are in regular expression format.  For example, if the incoming request is  <code>http://apigw/org/sandbox/petstore/base?petid=100&amp;display=detailed</code>  and you specify a white list entry of <code>^petid\$</code> , the target URL at run time will be  <code>http://myhost/mypath?storeid=3&amp;petid=100</code>  By default, Allowlist is selected, with no allowlist entries, meaning that <b>no</b> parameters are copied.	string
Stop on error	No	Select the errors that, if thrown during the policy execution, cause the assembly flow to stop. If there is a <b>catch</b> flow configured for the error, it is triggered to handle the error thrown. If an error is thrown and there are no errors selected for the Stop on error setting, or if the error thrown is not one of the selected errors, the policy execution is allowed to complete, and the assembly flow continues.	string
Response object variable	No	The name of a variable that will be used to store the response data from the request. By default, the invoke response, that is the body, headers, statusCode and statusMessage, is saved in the variable <code>message</code> . Use this property to specify an alternate location to store the invoke response. This variable can then be referenced in other actions, such as <a href="#">Map</a> .  Note: If you want the response to be saved in <code>message</code> , leave the Response object variable property blank, <b>do not</b> supply the value <code>message</code> .	string

6. Specify a version for the policy by clicking the Source icon , and completing the `version` section of the policy YAML. For example:

```
execute:
- invoke:
 version: 2.0.0
 title: invoke
 ...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

## Example

```
- invoke:
 version: 2.0.0
 title: get the account status
 target-url: https://example.com/accounts/{id}?status={status}
 cache-response: time-to-live
 cache-putpost-response: true
 tls-profile: MyTLSProfile
 verb: POST
 timeout: 60
 compression: false
 username: MyUser
 password: MyPassword
 stop-on-error:
 - ConnectionError
 - OperationError
```

DataPower Gateway (v5 compatible) only

# Configuring the Invoke policy for DataPower Gateway (v5 compatible)

Follow these steps to configure the Invoke policy for DataPower® Gateway (v5 compatible) in the assembly user interface.

## About this task

Note: This topic describes the Invoke policy implementation in the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway, see [Configuring the Invoke policy for DataPower API Gateway](#). For information about the different types of gateway, see [API Connect gateway types](#). For details on how to configure the policy in your OpenAPI source, see [invoke](#).

You might notice that the last invoke in their policy is replaced by a proxy. The replacement is sometimes done automatically by the IBM® API Connect DataPower Gateway to improve performance. The proxy is functionally equivalent to the invoke, but the API caller might notice the following differences when proxy is used.

- If the HTTP request made by the invoke or proxy gets a redirect (3xx) response:
  - invoke returns the response from following the redirect response.
  - proxy does not follow 3xx responses, and the redirect response is returned.
- The IBM API Connect test tool shows that proxy was used, but invoke appears in the Analytics latency records.
- The response from the proxy can contain different whitespace or escaping than a response from invoke. Despite the differences in the response, it is still valid.

If you want to prevent replacement of the last invoke in the assembly with proxy, you can set the API property `api.properties.x-ibm-gateway-optimize-invoke` to `false`. For more information, see [API properties](#).

## Procedure



1. In the navigation pane, click  Develop, then select the APIs tab. The Develop page opens.
2. Click the title of the API definition that you want to work with.
3. Select the Gateway tab, then click Policies in the navigation pane. For more information about working with the assembly editor for an API, see [The assembly editor](#).
4. Find the Invoke policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. Invoke policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is <code>invoke</code> .	string
Description	No	A description of the policy.	string
URL	Yes	Specifies a URL for the target service. For a SOAP API, a URL is added by default. Where possible, the Invoke URL value is pre-supplied from information that is defined in the imported WSDL.	string
TLS profile	No	Specifies a TLS profile to use for the secure transmission of data.	string
Timeout	Yes	The time to wait before a reply from the endpoint (in seconds). The default value is <code>60</code> .	integer
Follow redirects	No	Specifies the behavior if the back-end server returns the HTTP status code <b>301 Moved Permanently</b> . If you select this checkbox, the <code>invoke</code> policy follows the URL redirection by making a further call to the URL specified in the <code>Location</code> header in the response. If you clear this checkbox, the <code>invoke</code> saves the <b>301</b> status code and the API call is considered to be complete. Note: The <code>follow-redirect</code> property is supported only by the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), the <code>invoke</code> always follows the URL redirection; the <code>proxy</code> policy (not supported by the DataPower API Gateway) saves the <b>301</b> status code and completes the API call without following the URL redirection.	boolean
Username	No	The username to use for HTTP Basic authentication.	string
Password	No	The password to use for HTTP Basic authentication.	string
HTTP Method	Yes	The HTTP method to use for the Invoke. The following values are valid. <ul style="list-style-type: none"><li>• Keep</li><li>• GET</li><li>• POST</li><li>• PUT</li><li>• DELETE</li><li>• PATCH</li><li>• HEAD</li><li>• OPTIONS</li></ul> The default value is <code>GET</code> . However, if set to <code>Keep</code> , or the property is removed from the source, the HTTP method from the incoming request is used.	string
Compression	No	Select this checkbox to enable Content-Encoding compression on upload. The checkbox is cleared by default.	boolean

Property label	Required	Description	Data type
Cache Type	No	<p>The cache type determines whether to cache documents, honoring or overriding the HTTP Cache Control directives received in the response from the target URL. This property takes effect only when a response is received, otherwise the policy always returns the non-expired response that was previously saved in cache. Valid values are:</p> <p><b>Protocol</b> The cache behavior is determined by the Cache-Control headers on the response, in accordance with RFC 7234. To optimize performance, if the gateway receives more than one request for a resource that is not in the cache but could be cached when the response from the target URL is received, the gateway sends only one request to the target URL; the remaining requests are not processed until the response from the first request has been received and the cache behavior has been determined from this response. If the response indicates that caching is possible, the gateway responds to all waiting requests with the cached resource. If the response indicates that caching is not possible, the gateway sends all waiting requests to the target URL.</p> <p>Use this option only if you expect that responses from the target URL can be cached, in which case it should improve performance and limit the demand on the target URL. If, however, the target URL never indicates that the gateway should cache its response, performance might be impaired when compared to the No Cache option.</p> <p><b>No Cache</b> Responses from the target URL are not cached on the gateway regardless of any caching headers returned. In this case, every request from the client is sent to the target URL. Use this option if you do not want to cache any of the backend responses on the gateway, or if it is unlikely that a response from the target URL will allow caching through the Cache-Control header settings.</p> <p><b>Time to Live</b> This option is similar to the Protocol option except it allows you to specify the amount of time that you want the successful response from the invoke or proxy to remain in the cache. Use this option only if you expect that responses from the target URL can be cached.</p> <p>The default value is Protocol.</p>	string
Time to Live	No	Specifies the amount of time in seconds that the response stays in the cache. Applies only if the property Cache type is set to <b>Time to Live</b> . Enter a value in the range 5 - 31708800. The default value is <b>900</b> .	integer
Cache key	No	Specifies the unique identifier of the document cache entry. If omitted, the entire URL string is used as the key.	string
Stop on error	No	Select the errors that, if thrown during the policy execution, cause the assembly flow to stop. If there is a <b>catch</b> flow configured for the error, it is triggered to handle the error thrown. If an error is thrown and there are no errors selected for the Stop on error setting, or if the error thrown is not one of the selected errors, the policy execution is allowed to complete, and the assembly flow continues.	string
Response object variable	No	The name of a variable that will be used to store the response data from the request. By default, the invoke response, that is the body, headers, statusCode and statusMessage, is saved in the variable <i>message</i> . Use this property to specify an alternate location to store the invoke response. This variable can then be referenced in other actions, such as <a href="#">Map</a> . Note: If you want the response to be saved in <i>message</i> , leave the Response object variable property blank, <b>do not</b> supply the value <i>message</i> .	string

6. Specify a version for the policy by clicking the Source icon , and completing the **version** section of the policy YAML. For example:

```
execute:
- invoke:
 version: 1.0.0
 title: invoke
 ...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

## Example

```
- invoke:
 version: 1.0.0
 title: get the account status
 timeout: 60
 verb: POST
 cache-response: time-to-live
 cache-ttl: 900
 stop-on-error:
 - ConnectionError
 - OperationError
 tls-profile: MyTLSProfile
 target-url: https://example.com/accounts/{id}?status={status}
 username: MyUser
 password: MyPassword
```

## JSON to XML

Use the JSON to XML policy to convert the context payload of your API from the JavaScript Object Notation (JSON) format to the extensible markup language (XML) format.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

The JSON to XML policy uses a simple convention, based on BadgerFish, to convert your API context payload from JSON to XML. The policy expects the JSON input to be in the same format as the BadgerFish convention, so the structure can be rebuilt in XML. No additional configuration is required. For more information about the BadgerFish convention, see [BadgerFish](#).

Note: The JSON to XML policy does convert the JSON structure { "a" : "hello" } (which is not BadgerFish convention) into `<a>hello</a>`. Use the IBM® API Connect API Designer assembly view when you are creating your API definition to add a built-in policy to the flow.

The policy must be attached to the flow at the point at which you require the conversion to be performed. For example, if you need to convert a JSON-formatted request into an XML-formatted request, the policy must be attached to the request flow.

The policy reads input from the `message.body`, if that context exists, otherwise from the `request.body`, and then writes the output to the `message.body`.

Note: If you are using the DataPower API Gateway, the input to the JSON to XML policy must be parsed data. One way to produce parsed data is to use a [Parse](#) policy before a JSON to XML policy in your assembly flow, which provides explicit control of the parse action.

- [Configuring the JSON to XML policy for DataPower API Gateway](#)  
Follow these steps to configure the JSON to XML policy for DataPower API Gateway in the assembly user interface.
- [Configuring the JSON to XML policy for DataPower Gateway \(v5 compatible\)](#)  
Follow these steps to configure the JSON to XML policy for DataPower Gateway (v5 compatible) in the assembly user interface.



## Configuring the JSON to XML policy for DataPower API Gateway

Follow these steps to configure the JSON to XML policy for DataPower® API Gateway in the assembly user interface.

### About this task

Note: This topic describes the JSON to XML policy implementation in the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), see [Configuring the JSON to XML policy for DataPower Gateway \(v5 compatible\)](#). For more information about the different types of gateway, see [API Connect gateway types](#). For information about how to configure the policy in your OpenAPI source, see [json-to-xml](#).


### Procedure

1. In the navigation pane, click Develop, then select the APIs tab.  
The Develop page opens.
2. Click the title of the API definition that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.  
For more information about working with the assembly editor for an API, see [The assembly editor](#).
4. Find the JSON to XML policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. Policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is <code>json-to-xml</code> .	string
Description	No	A description of the policy.	string
Input	No	The input message to convert. Specify the name of a variable in the API context. <code>variableName.body</code> , the message payload, represents the JSON input to convert. The default value of the variable is <code>message</code> and <code>message.body</code> is the default input.	string
Output	No	The output message to store the conversion result. Specify the name of a variable in the API context. <code>variableName.body</code> represents the result of conversion from JSON format to XML format. When the specified input message is the default message, the default output is <code>message.body</code> . Otherwise, when the input message is the variable <code>my-message-variable</code> , for example, the default output is <code>my-message-variable.body</code> . The variable cannot be any read-only in the API context.	string
Conversion type	No	The conversion type that determines the target format of the output. The following options are available: <ul style="list-style-type: none"> <li>• None: No conversion of the output takes place.</li> <li>• badgerFish: BadgerFish convention is used to determine the target conversion format of the output.</li> </ul>	string
Root XML Element Name	Yes	The root element name of the resultant XML document. This property is used only if the input JSON document is not hierarchical and has more than one uppermost level property, or if the Always output the root element checkbox is selected. The default value is <code>json</code> .	string

Property label	Required	Description	Data type
Always output the root element	Yes	Select this checkbox if you always want the policy to output the root element, even if it is not required to make the XML document well formed. The default value is <b>false</b> .	boolean
Element name for JSON array elements	No	The XML element name to be used for JSON array elements.	string

6. Specify a version for the policy by clicking the Source icon , and completing the **version** section of the policy YAML. For example:

```
execute:
- json-to-xml:
 version: 2.0.0
 title: json-to-xml
...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

## Example

For example, the following simple JSON object

```
{ "a": { "$" : "hello" } }
```

becomes

```
<a>hello
```

The following JSON object with an attribute

```
{ "a": { "$" : "hello", "@type" : "world" } }
```

becomes

```
hello
```

DataPower Gateway (v5 compatible) only

## Configuring the JSON to XML policy for DataPower Gateway (v5 compatible)

Follow these steps to configure the JSON to XML policy for DataPower® Gateway (v5 compatible) in the assembly user interface.

### About this task

Note: This topic describes the JSON to XML policy implementation in the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway, see [Configuring the JSON to XML policy for DataPower API Gateway](#). For more information about the different types of gateway, see [API Connect gateway types](#). For details on how to configure the policy in your OpenAPI source, see [json-to-xml](#).

### Procedure



- In the navigation pane, click  Develop, then select the APIs tab.  
The Develop page opens.
- Click the title of the API that you want to work with, or create a new API.
- Select the Gateway tab, then click Policies in the navigation pane.  
For more information about working with the assembly editor for an API, see [The assembly editor](#).
- Find the JSON to XML policy in the palette, and drag the policy onto your canvas.
- Specify the following properties.

Table 1. Policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is <b>json-to-xml</b> .	string
Description	No	A description of the policy.	string
Root XML Element Name	Yes	The root element name of the resultant XML document. This property is used only if the input JSON document is not hierarchical and has more than one uppermost level property, or if the Always output the root element checkbox is selected. The default value is <b>json</b> .	string
Always output the root element	Yes	Select this checkbox if you always want the policy to output the root element, even if it is not required to make the XML document well formed. The default value is <b>false</b> .	boolean
Element name for JSON array elements	No	The XML element name to be used for JSON array elements.	string

6. Specify a version for the policy by clicking the Source icon , and completing the **version** section of the policy YAML. For example:

```
execute:
- json-to-xml:
```

```
version: 1.0.0
title: json-to-xml
...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

## Example

For example, the following simple JSON object

```
{ "a": { "$" : "hello" } }
```

becomes

```
<a>hello
```

The following JSON object with an attribute

```
{ "a": { "$" : "hello", "@type" : "world" } }
```

becomes

```
hello
```



## Log

Use the Log policy to customize or override the default activity logging configuration for an API.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.3.0 or later)

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [log](#).

## About

By default, the logging of activity when an API is called, and the sending of the log data to the analytics server, is determined by the settings in the API definition, as described in [Configuring activity logging](#) (OpenAPI 2.0) or [Configuring activity logging](#) (OpenAPI 3.0). However, you can customize activity logging by adding a Log policy to the API assembly flow.

The Log policy enables you to gather all API analytics data for processing in your assembly, and to control what activity log data is sent to the analytics server. Here are some examples of the ways in which you can use the Log policy to customize API activity logging:

- Gather activity log data, use a Redaction policy to redact sensitive fields in the data, then send to the analytics server.
- Send different levels of analytics data depending on certain conditions; for example, the API operation that was called.
- Configure custom activity logging in a global policy that is applied to every API in a Catalog. For more information on global policies, see [Working with global policies](#).

## Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Log policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is <b>log</b> .	string
Description	No	A description of the policy.	string

Property label	Required	Description	Data type
Mode	Yes	<p>Select one of the following options:</p> <ul style="list-style-type: none"> <li>Gather-only: gather all analytics data and write it to the <code>log</code> context variable, which populates the API event record on completion of the API execution. For more information on the fields in the <code>log</code> context variable, and the consequent API event record, see <a href="#">API event record fields</a>.</li> <li>Send-only: perform the following actions: <ul style="list-style-type: none"> <li>Read the data from the <code>log</code> context variable.</li> <li>Truncate all message payloads and convert to a textual representation.</li> <li>Send the data to the analytics server.</li> </ul> </li> <li>Gather-and-send: perform a <b>gather-only</b> operation, immediately followed by a <b>send-only</b> operation.</li> </ul> <p>If you use the Send-only or Gather-and-send option, data is buffered and sent to the analytics server in batches according to the time interval configured for the Analytics Endpoint on the DataPower API Gateway. For more information, see <a href="#">Configuring an analytics endpoint</a> in the DataPower knowledge center.</p> <p>Note: If you are offloading to a third party analytics server, you can redact any aspect of the event data. If you are using API Connect analytics, you can redact only request and response payloads.</p>	string
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> Log Level	No	<p>The type of content to log. Enter one of the following values:</p> <ul style="list-style-type: none"> <li><code>none</code>: Indicates no logging.</li> <li><code>activity</code>: Logs the invocation only, which is only the resource URI.</li> <li><code>header</code>: Logs activity and header.</li> <li><code>payload</code>: Logs activity, header, and payload.</li> <li><code>default</code>: Use the log level setting defined in the API definition; for more information, see <a href="#">Configuring activity logging</a> (OpenAPI 2.0) or <a href="#">Configuring activity logging</a> (OpenAPI 3.0). This is the default value.</li> <li><code>\$(value)</code>: An inline parameter in the format <code>\$(value)</code> to retrieve a value from the API context.</li> </ul>	

## Map

Use the Map policy to apply transformations to your assembly flow and specify relationships between variables.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.5.0.0 or later)

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [map](#).

## About

Be aware that MAP decodes all parameter values to an ASCII character equivalent. If a MAP parameter contains entries such as `%nn`, the MAP output contains decoded values.

For information about the structure of a Map policy and its behavior, see [The Map policy structure](#).

For information about configuring a Map policy by using the user interface, see [Configuring the Map policy in the user interface](#).

For examples of YAML representations of different Map policy configurations, see [Map policy examples](#).

## Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Map policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The field is visible when editing inputs. The default value is <code>map</code> .	string
Description	No	A description of the policy. The field is visible when editing inputs.	string
Inputs	Yes	A list of variables that are inputs of the policy.	array (string)
Outputs	Yes	A list of variables that are outputs of the policy.	array (string)
Value	Yes	A GatewayScript program to be performed by the policy in order to map its inputs to its outputs, or to set the value of outputs.	string

Note: The map policy has other properties that are not displayed in the user interface. For a complete list of properties, see [map](#).

- [The Map policy structure](#)

The Map policy uses a structure within its OpenAPI definition to specify the behavior of the policy.

- [Configuring the Map policy in the user interface](#)

The assemble view in API Designer provides a visual representation of the relations between the inputs and outputs of your Map policy.



- [Map policy examples](#)  
Examples of the OpenAPI definitions of Map policies.

## Related tasks

---

- [Creating a new REST OpenAPI definition](#)

## The Map policy structure

The Map policy uses a structure within its OpenAPI definition to specify the behavior of the policy.

This topic contains the following sections:

- [Structure](#)
- [Input and output definitions](#)
- [Actions](#)
- [Script](#)
- [Fields](#)
- [References to inputs and outputs](#)
- [Accessing other contexts](#)

For map policy examples, see [Map policy examples](#).

Note: If you deploy your API to the DataPower® Gateway (v5 compatible) then, with the exception of client ID and client secret, the passing of form input as a parameter into an API is not supported. This restriction does not apply if you deploy your API to the DataPower API Gateway.

## Structure

---

In addition to its title and description, a Map policy has the following four main sections:

### inputs

A list of variables that form the input of the Map policy. Each input has a context variable in which the input variable is found, the variable's name within the Map policy, the content type of the variable, and the definition of the variable or a schema defining its structure.

### outputs

A list of variables that form the output of the Map policy. These include a context variable where the output variable is found or should be created, the variable's name at output, and the definition of the variable or a schema defining its structure.

### action

An array containing details of the actions to be performed in order. Each entry includes either a **set** or **create** field, which specifies the output variable or variables that are part of the action. An action can also contain a **from** field, which specifies the input variable or variables that are part of the action.

Each action also contains either a **value** field, when the output is set or created, or a **foreach** field and a new **actions** section, when further actions are nested within the first to set or create elements in a nested array.

### options

The following property options are available for you to select:

- Include empty. If the check box is selected (the default option), empty XML elements are included in the output of the map policy. Clear the check box if you do not want empty XML elements to be included in the output of the map policy.
- Namespace inheritance. If the check box is selected (the default option), XML namespaces are inherited from the parent element. Clear the check box if you want the map policy to use explicit namespaces.
- Namespace inlining. If the check box is selected (the default option), XML namespaces will be inserted into the document where they are first used. Clear the check box if you want namespaces to all be defined on the root element.
- API Gateway only Resolve XML input data type. If the check box is selected, XML elements whose schema is configured as type boolean or numeric will be converted to that data type. Clear the check box (the default option) if you want all XML element values to be returned as a string.
- API Gateway only Parse XML output: If the check box is selected, the map policy will write XML output to **message.body** as a parsed XML document. By default, the XML will be output as an XML string. XML output to any other variable will always be written as an XML string.
- API Gateway only Empty XML element handling. This property controls how the map policy handles the output of an empty XML element. The following choices are available:
  - String (the default option): An empty XML element is handled as an empty string.
  - Null: An empty XML element is handled as a null value.
  - None: The data is ignored.
  - API Gateway only String Badgerfish: The value for an empty XML element is considered to be an empty string. The empty string value will be placed into a JSON badgerfish value property.
  - API Gateway only Null Badgerfish: The value for an empty XML element is considered to be null. The null value will be placed into a JSON badgerfish value property. A mapping of this element to a JSON output property does not occur unless the Allow null value option is selected.
- API Gateway only Use only first array element. If the check box is selected, if an array is encountered in the traversal of the input, only the first element is used. Clear the check box (the default option) if you want the map policy to use all array elements.
- API Gateway only Resolve API Connect variable references. If the check box is cleared (the default option), the map policy ignores API Connect variable references in the map properties. Select the check box if you want API Connect variable references found in the map properties to be resolved.
- API Gateway only Allow null value. If the check box is selected, an input property value with a null value is mapped to the output document. Clear this check box (the default option) if you want the map policy to ignore null input values.

- API Gateway only** Optimize schema definition. If the check box is selected, complex schema types evaluation handles circular type references in an optimized manner. Clear this check box (the default option) to evaluate these schema types in a standard manner.
- API Gateway only** Create required child properties for array objects and mapped optional objects. If the check box is selected, default values are generated in the output for required properties that are either not mapped, or for which there is no input data present, in the following specific cases:
  - An array consists of objects that contain one or more required properties.
  - An object which is optional has one or more child properties that are required.

By default, these required properties are not present in the output. If you select this option, these required properties will be present in the output. If the output schema defines a **default** property for the output property then the specified default value is used, otherwise a default value is assigned dependent on the data type, as follows:

- String: empty string ("")
- Number: 0
- Boolean: false
- Object: empty object
- Array: empty array

#### Example 1

The input data has the following array of objects:

```
[{"a": "value1"}, {"a": "value2", "b": "value3"}]
```

The output schema defines the output object as having two properties, **a** and **b**, of which **b** is required. The map policy defines the following mappings:

- `input.array.a` to `output.array.a`
- `input.array.b` to `output.array.b`

If the check box is selected, and **b** is either not mapped or has no input data present, then **b** is assigned a default value of an empty string, and the output is as follows:

```
[{"a": "value1", "b": ""}, {"a": "value2", "b": "value3"}]
```

#### Example 2

The output schema defines the following structure:

```
{"a" : {"b" : {"c" : "value1", "d" : "value2"} } }
```

Property **b** is optional but property **d** within **b** is required.

The map policy defines a mapping to `output.a.b.c`.

If the check box is selected, and **d** is not mapped, then **d** is assigned a default value of an empty string, and the output is as follows:

```
{"a" : {"b" : {"c" : "value1", "d" : ""} } }
```

If the check box is not selected, these required properties are **not** created in the output with their default values.

- API Gateway only** Empty JSON array handling. This property controls how the map policy handles the output of an empty array. The following choices are available:
  - All (The default value): Output all empty arrays, including empty children arrays.
  - Parent: Output only the current property's empty array value. Children map actions of this property are not attempted.
  - None: Prevent any empty output array values from being produced.
- API Gateway only** Enable JSON post processing. Select this check box to enable post processing of mapped JSON output. The post processing of JSON output will use the output schema to ensure that property values are of the same data type as that defined in the schema. It will also normalize output property values that have a Badgerfish JSON syntax due to object mapping of an XML input. The check box is cleared by default, meaning that there is no post processing of mapped JSON output.
- API Gateway only** Schema definition circular reference limit. Set the value of this property to an integer value that specifies the maximum allowed number of iterations of a circular schema definition. The default value is 1, which means that circular schema definitions are not followed. The maximum possible value is 5.
- Severity level for input data log messages; This property specifies the severity level for log messages that relate to input data. The following choices are available:
  - error
  - warn
  - info
- API Gateway only** Create empty parent object for failed mapping. If a mapping fails because its input is not present and there is no default mapping configured, the default behavior is to not to make any change to the output mapping. Select this check box to create an empty object for the parent of the target mapping, emulating the behavior of IBM® API Management Version 4.0.

#### Example

The map policy defines a mapping to `output.a.b.c`.

If input data is present, the output is as follows:

```
{
 "a": {
 "b": {
 "c": "inputvalue"
 }
 }
}
```

If there is no input data, and the Create empty parent object for failed mapping option is selected, the output is as follows:

```
{
 "a": {
 "b": {
```

```
}
}
}
```

Properties **a** and **b** are created but the value of **b** is an empty object.  
The check box is cleared by default.

## Input and output definitions

You define the inputs and outputs of your Map policy in their own sections. Each input or output is an element in the array `inputs` or `outputs` and is defined by a name, a schema definition or reference, and a variable within the context from which it should be read or to which it should be written. After they have been defined, inputs and outputs are referenced by the name provided in the definition, not by the name of the variable.

The following example shows the inputs and outputs sections of a Map policy.

```
inputs:
 input_string:
 schema:
 type: string
 variable: request.parameters.name_in
 input_integer:
 schema:
 type: integer
 variable: request.parameters.age_in
outputs:
 output:
 schema:
 $ref: '#/definitions/output'
 variable: message.body
```

The `schema` field specifies the schema that describes the variable and can be a simple type, a reference to a definition, or an inline schema definition.

The `variable` field describes the variable and the context that should be assigned to the input or output variable during the execution of the map policy.

## Actions

The fields included in the `actions` section are used in the following ways:

### set

Use the `set` field when you want to assign the result of the `value` field to the output variable specified in the `set` field, replacing the existing value of the output variable. You can specify only one output variable, although this variable can be an array or object.

### create

Use the `create` field when you want to use the result of the `value` field to create a new entry for the output array specified in the `create` field, appending it to the array. You can specify only one output variable, although this variable can be an array or object.

### from

Specify which variables are used in the action as either a single variable or an array of variables, where a variable can be an array or an object. The `from` field is not included if no inputs are used.

### value

Use GatewayScript to provide a script that produces output variables. When a single input is mapped to a single output, the `value` field can be omitted and the variable in `from` is set or created as the variable in `set` or `create` respectively.

### default

Provide a static value, or an inline variable reference, to be applied to the output when no input value is provided. For information on inline variable references, see [Inline references](#).

### foreach

Specify a variable if you want to execute the associated `actions` field for each entry of the array. The variable can be from the input or output of the Map policy.

### actions

Use the `actions` field to nest actions within an action. Because another action could achieve the same result if applied only once, it is primarily for use with the `foreach` field.

## Script

In a `value` or `default` field, use GatewayScript to write the behavior of the action to which the `value` field belongs.

Include the script in single quotation marks. For example: `'4 + 5'` or `'variable_1.toUpperCase()'`.

For information about GatewayScript, see [Gateway programming model and GatewayScript](#)

## Fields

### from, set, and create

Each action must have a single `set` or `create` field that specifies the output variable to which the action is applied. Each action can also have a `from` field containing one or more entries that are used to specify the input variable or variables that are used in the action.

The `set` and `create` fields are both used to assign values to the output variable.

- `set` replaces the current value of the output variable or creates the variable if it does not already exist.
- `create` appends a new array entry to the output variable.

For `set` and `create`, use `output_variable_name.variable_name` to specify which of your defined output variables to use, where `output_variable_name` is as defined in the outputs section of the Map policy and `variable_name` refers to an optional field that belongs to the output variable.

For `from`, use `input_variable_name.variable_name` to specify which of your defined output variables to use, where `input_variable_name` is as defined in the outputs section of the Map policy and `variable_name` refers to an optional field that belongs to the output variable.

### foreach

Use the **foreach** field to specify an input array for which the following **actions** or **value** field will be executed for each entry of the array.

For example:

```
foreach: input.in
actions:
 actions
```

where *input.in* is an input variable that is an array and *actions* is one or more actions in the same format as the parent section. In this example the instructions specified in *actions* are executed once for each array entry of *input.in*.

Referencing the input variable specified in the **foreach** field references the array entry that the current iteration corresponds to.

If a single variable is used in the **foreach** field instead of an array, the following **actions** or **value** field will be applied to or based upon the single variable once and then the loop will terminate.

## References to inputs and outputs

Reference variables from the **from** field either by name or by using a number preceded by a '\$' and enclosed in parentheses. The variables are numbered from 1, where 1 is the first variable in the array, or the only variable when the **from** field lists only a single variable. For example:

```
value: '$(1) + $(2)'
```

or

```
value: '$(variable_1) + $(variable_2)'
```

where each *variable* is a variable that is included in the **from** field.

During a **foreach** loop, you can reference **\$(0)**. The **\$(0)** variable begins a **foreach** loop empty but, after an iteration, becomes equivalent to the output of the iteration and can then be referenced again. In this manner, you can apply an array to a single value. For example:

```
- set: out.total
from: in.input
foreach: in.input
value: '$(0) + $(in.input)'
```

where *out.total* is referenced by **\$(0)**. In each iteration, the current value of *out.total* and the current array entry of *in.input* are summed, and the value of *out.total* is set as this summation.

When using a **foreach** to operate on an array, if the elements of the array do not have named fields, you can use **\$(this)** to reference the current level of nesting.

## Accessing other contexts

At any point within your Map policy's **value** or **default** fields, you can access the context of the API call using the syntax **\$(context.variable)**.

Alternatively, you can include the variables from other contexts when you define an input to your map policy and then reference it as you would any other input variable.


For a list of available context variables, see [API Connect context variables](#).

## Configuring the Map policy in the user interface


The assemble view in API Designer provides a visual representation of the relations between the inputs and outputs of your Map policy.






### Procedure

To configure your Map policy, complete the following steps:

1. Click the Map policy in the canvas of the Assemble view.  
The property sheet opens.
2. Click the Edit inputs icon  in the Input column.
3. Add an input variable.
  - a. Click + input.
  - b. In the Context variable field for an input, provide the location of your input variable in the context of the assembly. For a list of context variables, see [API Connect context variables](#).
  - c. In the Name field for an input, provide a name for your input for use within only the Map policy.  
Note:  
You must ensure that the name that you provide does not exactly match the value of the Context variable field, or the result might be unpredictable.
  - d. Optional: In the Content type field, specify the type of your input. If None is selected, the content type is treated as JSON.
  - e. In the Definition field for an input, provide the type of the variable.  
The type can be one from a standard set of types, a definition that you have created for your API, or you can select Inline schema to provide a schema as one of the following options:
    - YAML
    - JSON
    - Generate from sample JSON
    - Generate from sample XML

If you select Object or Array, you can create a schema through the user interface after you have clicked Done and returned to the main view of the property sheet.


4. Optional: To remove a variable, click the corresponding Remove input icon .
5. After you have added all of your input variables, click Done.


6. Click the Edit outputs icon  in the Output column.
7. Add an output variable.
  - a. Click + output.
  - b. In the Context variable field for an output, provide the location of your output. This location can be a new context or one already established during the assembly. For a list of context variables, see [API Connect context variables](#).
  - c. In the Name field for an output, provide a name for your variable to use within the Map policy and when it is included in its context at output.  
Note:  
You must ensure that the name that you provide does not exactly match the value of the Context variable field, or the result might be unpredictable.
  - d. Optional: In the Content type field, specify the type of your input. If None is selected, the content type is treated as JSON.
  - e. In the Definition field for an output, provide the type of the variable.  
The type can be one from a standard set of types, a definition that you have created for your API, or you can select Inline schema to provide a schema as one of the following options:
    - YAML
    - JSON
    - Generate from sample JSON
    - Generate from sample XML
 If you select Object or Array, you can create a schema through the user interface after you have clicked Done and returned to the main view of the property sheet.
8. Optional: To remove a variable, click the corresponding Remove output icon .
9. After you have added all of your output variables, click Done.
10. Optional: If you selected Object or Array for the type of an input or output, create an inline schema definition through the user interface by completing the following steps:
  - a. For an Array, click add item. Provide a type for the item and then click the Add icon .
  - b. For an Object, click add property. Provide a name and type for the property and then click the Add icon .
 For objects and arrays created in this manner, you can continue to add items and properties, which can themselves be objects and arrays.
11. To connect an input variable to an output variable, click the circle that is directly before the input variable and then click the circle that is directly before the output variable.  
A green line is drawn, linking the two variables together. You can connect multiple inputs to a single output, and a single input can be connected to multiple outputs.
12. To configure an output, whether it has inputs connected to it or not, click the circle directly before the output variable without first clicking on a circle for an input variable.  
The Configure mapping window opens.
13. Optional: In the Mapped from section of the window, you can view which inputs are mapped to the output you are editing. To remove an input, click the Remove input icon  alongside the input.  
If the output is part of an array, further configuration options are available. The array, or levels of array in the case of a multidimensional array, can be created by iterating arrays on the input side of the mapping. For each level of your array, select which array on the input side is to be iterated. In the Value field, you can use `$(this)` to reference elements of an array that are not named within the array.
14. Optional: In the Value field, use GatewayScript to configure how any inputs are transformed to produce the output.  
For more information about valid code, see the [Script](#) section of the [The Map policy structure](#) topic.
15. Optional: In the Default field, provide a static value, or an inline variable reference, to be applied to the output when no input value is provided. For information on inline variable references, see [Inline references](#).
16. Optional: To delete all mappings to the output, click Delete.
17. When you have configured your outputs, click OK.
18. Optional: Click the Settings icon in the Map column.
  - a. Optional: Provide a Title and Description for your Map policy.
  - b. To control the XML output of the map policy, select the following options as required:
 




**Include empty**  
If the check box is selected (the default option), empty XML elements are included in the output of the map policy. Clear the check box if you do not want empty XML elements to be included in the output of the map policy.

**Namespace inheritance**  
If the check box is selected (the default option), XML namespaces are inherited from the parent element. Clear the check box if you want the map policy to use explicit namespaces.

**Namespace inlining**  
If the check box is selected (the default option), XML namespaces will be inserted into the document where they are first used. Clear the check box if you want namespaces to all be defined on the root element.

 **Resolve XML input data type**  
If the check box is selected, XML elements whose schema is configured as type boolean or numeric will be converted to that data type. Clear the check box (the default option) if you want all XML element values to be returned as a string.

 **Parse XML output**  
If the check box is selected, the map policy will write XML output to `message.body` as a parsed XML document. By default, the XML will be output as an XML string. XML output to any other variable will always be written as an XML string.

Note: These options effect the XML output only and have no effect on the JSON data.
- c.  From the Empty XML element handling list, select one of the following options to control how the map policy handles the output of an empty XML element:
  - String (the default option): An empty XML element is handled as an empty string.
  - Null: An empty XML element is handled as a null value.
  - None: The data is ignored.
  -  String Badgerfish: The value for an empty XML element is considered to be an empty string. The empty string value will be placed into a JSON badgerfish value property.
  -  Null Badgerfish: The value for an empty XML element is considered to be null. The null value will be placed into a JSON badgerfish value property. A mapping of this element to a JSON output property does not occur unless the Allow null value option is selected.

d. **API Gateway only** Select the following general configuration options as required:

Use only first array element

If the check box is selected, then if an array is encountered in the traversal of the input, only the first element is used. Clear the check box (the default option) if you want the map policy to use all array elements.

Resolve API Connect variable references

If the check box is cleared (the default option), the map policy ignores API Connect variable references in the map policies. Select the check box if you want API Connect variable references found in the map properties to be resolved.

Allow null value

If the check box is selected (the default option), an input property value with a null value is mapped to the output document. Clear this check box (the default option) if you want the map policy to ignore null input values.

Optimize schema definition

If the check box is selected, complex schema types evaluation handles circular type references in an optimized manner. Clear this check box (the default option) to evaluate these schema types in a standard manner.

**API Gateway only** Enable JSON post processing

If the check box is selected, post processing of mapped JSON output is enabled. The post processing of JSON output will use the output schema to ensure that property values are of the same data type as that defined in the schema. It will also normalize output property values that have a Badgerfish JSON syntax due to object mapping of an XML input. The check box is cleared by default, meaning that there is no post processing of mapped JSON output.

**API Gateway only** Create empty parent object for failed mapping

**API Gateway only** Use this setting to control the behavior if a mapping fails because its input is not present and there is no default mapping configured.

The default behavior is to make no change to the output, but if you select this check box an empty object is created for the parent of the target mapping, emulating the behavior of IBM API Management Version 4.0.

Example

The map policy defines a mapping to `output.a.b.c`.

If input data is present, the output is as follows:

```
{
 "a": {
 "b": {
 "c": "inputvalue"
 }
 }
}
```

If there is no input data, and the Create empty parent object for failed mapping option is selected, the output is as follows:

```
{
 "a": {
 "b": {
 }
 }
}
```

Properties `a` and `b` are created but the value of `b` is an empty object.

The check box is cleared by default.

Create required child properties for array objects and mapped optional objects

If the check box is selected, default values are generated in the output for required properties that are either not mapped, or for which there is no input data present, in the following specific cases:

- An array consists of objects that contain one or more required properties.
- An object which is optional has one or more child properties that are required.

By default, these required properties are not present in the output. If you select this option, these required properties will be present in the output. If the output schema defines a `default` property for the output property then the specified default value is used, otherwise a default value is assigned dependent on the data type, as follows:

- String: empty string ("")
- Number: 0
- Boolean: false
- Object: empty object
- Array: empty array

Example 1

The input data has the following array of objects:

```
[{"a": "value1"}, {"a": "value2", "b": "value3"}]
```

The output schema defines the output object as having two properties, `a` and `b`, of which `b` is required. The map policy defines the following mappings:

- `input.array.a` to `output.array.a`
- `input.array.b` to `output.array.b`

If the check box is selected, and `b` is either not mapped or has no input data present, then `b` is assigned a default value of an empty string, and the output is as follows:

```
[{"a": "value1", "b": ""}, {"a": "value2", "b": "value3"}]
```

Example 2

The output schema defines the following structure:

```
{"a" : {"b" : {"c" : "value1", "d" : "value2"} } }
```

Property `b` is optional but property `d` within `b` is required.

The map policy defines a mapping to `output.a.b.c`.

If the check box is selected, and `d` is not mapped, then `d` is assigned a default value of an empty string, and the output is as follows:

```
{ "a" : { "b" : { "c" : "value1", "d" : "" } } }
```

If the check box is not selected, these required properties are **not** created in the output with their default values.

- e. **API Gateway only** From the Empty JSON array handling list, select one of the following options to control how the map policy handles the output of an empty array:
- All (the default option): Output all empty arrays, including empty children arrays.
  - Parent: Output only the current property's empty array value. Children map actions of this property are not attempted.
  - None: Prevent any empty output array values from being produced.
- f. From the Severity level for input data log messages list, select one of the following options to specify the severity level for log messages that relate to input data:
- error
  - warn
  - info
- g. **API Gateway only** Set the Schema definition circular reference limit field to an integer value that specifies the maximum allowed number of iterations of a circular schema definition. The default value is 1, which means that circular schema definitions are not followed. The maximum possible value is 5.
- h. Click Save when done.

## Results

You have configured a Map policy to transform and map variables in your assembly flow.

## Map policy examples

Examples of the OpenAPI definitions of Map policies.

- [One to one mapping](#)
- [Many to one mapping](#)
- [A simple transformation using the value field](#)
- [Mapping from multiple contexts into a new context](#)
- [Mapping to an inline schema definition](#)
- [Mapping with a default value](#)
- [Mapping an array into a single value](#)
- [Mapping array elements to an array](#)
- [Using the advanced XML options](#)

## One to one mapping

The following example:

- maps parameters from the request's query to an object in the message body.
- maps one strings directly to a single string.
- maps one integer directly to a single integer.

The referenced definition, `output`, defines an object containing a string, name, and an integer, age.

```
- map:
 title: 1-1 map
 inputs:
 input_string:
 schema:
 type: string
 variable: request.parameters.name_in # the location of the variable, named "name_in" and found in the query
parameters of the request
 input_integer:
 schema:
 type: integer
 variable: request.parameters.age_in # another variable in the query parameters of the request
 outputs:
 output:
 schema:
 $ref: '#/definitions/output' # a schema definition reference to use for the output. The schema is for the
whole of the message body.
 variable: message.body
 actions:
 - set: output.name_out # in the actions section, variables are referenced by their name within the map policy
 from: input_string # if a value field is not used, the mapping is direct with the input value used for the
output variable
 - set: output.age_out # because the 'output' variable is itself an object, further references are made to
variables that it contains
 from: input_integer
```

## Many to one mapping

The following example:

- maps parameters from the request's query to an object in the message body.
- maps two strings to a single string by concatenating them.
- maps two integers to a single integer by summing them.

The referenced definition, `output`, defines an object containing a string and an integer.

```
- map:
 title: many-1 map
 inputs:
 input_string_1:
 schema:
 type: string
 variable: request.parameters.first_name
 input_string_2:
 schema:
 type: string
 variable: request.parameters.last_name
 input_integer_1:
 schema:
 type: integer
 variable: request.parameters.balance_1
 input_integer_2:
 schema:
 type: integer
 variable: request.parameters.balance_2
 outputs:
 output:
 schema:
 $ref: '#/definitions/output'
 variable: message.body
 actions:
 - set: output.full_name
 from:
 - input_string_1
 - input_string_2
 value: |
 var retValue = undefined;
 if ($(input_string_1) !== undefined && $(input_string_2) !== undefined) {
 retValue = $(input_string_1).toUpperCase() + ' ' + $(input_string_2).toUpperCase()
 }
 retValue;
 - set: output.total_balance
 from:
 - input_integer_1
 - input_integer_2
 value: |
 var i1 = 0;
 var i2 = 0;
 if ($(input_integer_1) !== undefined) i1 = $(input_integer_1);
 if ($(input_integer_2) !== undefined) i2 = $(input_integer_2);
 i1 + i2;
```

## A simple transformation using the value field

The following example:

- maps a parameter from the request's query to an object in the message body.
- maps one string featuring lowercase characters to a single string containing only uppercase characters.

The referenced definition, `output`, defines an object containing a single string.

```
- map:
 title: Uppercase map
 inputs:
 input_lowercase:
 schema:
 type: string
 variable: request.parameters.name_in
 outputs:
 output_uppercase:
 schema:
 $ref: '#/definitions/output'
 variable: message.body
 actions:
 - set: output_uppercase.name_out
 from: input_lowercase
 value: $(input_lowercase).toUpperCase() # the input variable is referenced and calls the toUpperCase method to
 produce a value for the output variable
```

## Mapping from multiple contexts into a new context

The following example:

- maps an integer from the request's header to an integer in a new message body.
- maps a string from the message's body to the body of a new custom context, named `new_context`.

```
- map:
 title: Context map
 inputs:
 input_integer:
 schema:
```



```

 type: integer
 variable: request.headers.age_in # the 'age_in' header of the request is used as an input
input_string:
 schema:
 type: string
 variable: message.body.name_in # as is the 'name_in' field of the request body
outputs:
 output_integer:
 schema:
 type: integer
 variable: message.body.age_out
 output_string:
 schema:
 type: string
 variable: new_context.body.name_internal # the context named 'new_context' is created by the map policy and
exists only while the assembly is processed
 actions:
 - set: output_string
 from: input_string
 - set: output_integer
 from: input_integer

```

## Mapping to an inline schema definition

The following example:

- maps parameters from the request's headers to an object in the message body, which is defined within the map policy.
- maps one string directly to a single string.
- maps one integer directly to a single integer.

```

- map:
 title: Inline schema map
 inputs:
 input_integer:
 schema:
 type: integer
 variable: request.headers.age_in
 input_string:
 schema:
 type: string
 variable: request.headers.name_in
 outputs:
 output:
 schema: # instead of a simple type or a reference to a definition, an inline YAML definition
is used
 type: object
 properties:
 name_out:
 type: string
 name: name_out
 age_out:
 type: integer
 format: int32
 name: age_out
 title: output
 variable: message.body
 actions:
 - set: output.age_out
 from: input_integer
 - set: output.name_out
 from: input_string

```

## Mapping with a default value

The following example:

- maps one string directly to a single string.
- provides a default value for the output string if a valid input string is not provided.

```

- map:
 title: Default Map
 inputs:
 input_string:
 schema:
 type: string
 variable: request.headers.name_in
 outputs:
 output_string:
 schema:
 type: string
 variable: message.body.name_out
 actions:
 - set: output_string
 from: input_string
 default: John Smith # the default field is specified in the same way as a value, in this case
providing a fixed value

```

## Mapping an array into a single value

The following map policy:

- maps a single array of integers into a single integer.
- sums the integers in the array.
- `$(0)` represents the accumulated output because map evaluates all array element values.

```

- map:
 title: Summation map
 inputs:
 input:
 schema:
 $ref: '#/definitions/balance_array_in'
 variable: request.body
 outputs:
 output:
 schema:
 type: integer
 variable: message.body.total_balance_out # instead of using a full schema definition of the message body, a
single variable in the message body is specified
 actions:
 - set: output
 from: input
 foreach: input # the foreach field specifies that each value of the array 'input' is to be iterated over
 value: $(0)+$(input) # the $(0) reference is the accumulated value of the 'output' variable

```

## Mapping array elements to an array

The following map policy:

- maps an array, whose elements are objects containing two integers, to an array, whose entries contain a single integer field.
- maps an array to an array of the same length.
- takes the difference of the values of the integers in each array element to create a single integer in each array element.

```

- map:
 title: Array summation
 inputs:
 input:
 schema:
 $ref: '#/definitions/balance_and_credit_array'
 variable: request.body
 outputs:
 output_array:
 schema:
 type: array
 variable: message.body
 actions:
 - create: output_array
 from: input
 foreach: input
 actions:
 - set: total_balance_out
 from: # inside the actions section, variables inside the array elements being
iterated over are used
 - integer_in_1
 - integer_in_2
 value: $(integer_in_1)-$(integer_in_2) # the difference of the variables is taken for each array entry
of 'input'

```

## Using the advanced XML options

The following example:

- does not include empty elements in the output of the map policy.
- uses explicit name spaces rather than inheriting name spaces from the parent element
- indicates that all namespace declarations will be placed on the root XML element.
- uses all array elements rather than only the first
- ignores API Connect variable references
- maps input property values with a null value to the output document
- handles circular type references in an optimized manne

```

actions:
 - set: output, two
 from: input, one
options:
 includeEmptyXMLElements: false
 namespaceInheritance: false
 inlineNamespaces: false
 mapArrayFirstElementValue: false
 mapResolveApicVariables: false
 mapNullValue: true
 mapOptimizeSchemaDefinition: true
 mapCreateEmptyArray: parent

```

## OAuth

An OAuth policy performs the requested OAuth processing based on the defined OAuth provider settings.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [oauth](#).

## About

By adding an assembly OAuth policy, you specify the OAuth provider settings to use to perform the requested OAuth processing and the supported OAuth processing components.

Note: You add an OAuth policy to the assembly in a native OAuth provider. For more information, see the following topics:

- [Editing the native OAuth provider configuration using the API Editor](#) (Cloud Manager UI)
- [Editing the native OAuth provider configuration using the API Editor](#) (API Manager UI)

You can specify the settings through one or more of the following methods.

- URL reference
- Literal configuration
- Object reference

When you use more than one method to specify the same aspect of the OAuth provider settings, the following precedence rules apply to enforce the settings dynamically for the incoming transactions.

- URL reference takes precedence over any existing literal configuration or object reference.
- Literal configuration takes precedence over any existing object reference.

You can configure the policy to support one or more of the following processing components:

Validate request

Validates the authorization request from the client.

Generate authorization code

Generates the authorization code for the client, which represents the resource owner's authorization that grants access to the requested resource.

Verify authorization code

Verifies the authorization code from the client.

Verify refresh token

Verifies the refresh token that is presented by the client.

Generate access token

Generates the access token to the client when the authorization code or refresh token is verified.

Introspect token

Introspects the token to determine its state and, when active, its metadata.

When the policy does not support a processing component but that processing is requested, the unsupported component is not run.

You can include multiple OAuth policies in your OAuth provider assembly. For example, your assembly might include the following flow:

1. OAuth policy that validates the request.
2. GatewayScript policy.
3. OAuth policy that generates authorization code.

For more information, see [Example - using multiple OAuth policies in an OAuth provider assembly](#) and [OAuth context variables](#).

## Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. OAuth policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is <code>oauth</code> .	string
Description	No	A description of the policy	string
Default OAuth Provider Settings Object	Yes	The name of an existing OAuth provider that defines the required settings.	string
Dynamic OAuth configuration from a URL	No	A URL to a document that contains serialized XML or JSON properties that defines OAuth token generate settings.	string
Dynamic OAuth configuration from a literal string	No	A literal string that contains serialized XML or JSON properties that defines OAuth token generate settings.	string
Supported OAuth components	No	Select the OAuth components that are supported by this policy.	string

## Overriding the default OAuth provider settings

You can use either the Dynamic OAuth configuration from a literal string property or the Dynamic OAuth configuration from a URL property to dynamically override any OAuth provider configuration settings defined by the Default OAuth Provider Settings Object property.

For example, to override the access token expiration time with a value of 200 seconds, include the following configuration in either the literal string or the document at the specified URL:

```
<OAuthProviderSettings><APICAccessTokenTTL>200</APICAccessTokenTTL></OAuthProviderSettings>
```

For a list of all OAuth provider settings, refer to the `OAuthProviderSettings` management schema, defined in the `xml-mgmt.xsd` file located in the store: directory on the DataPower API Gateway.

If you are using the API Manager user interface, the connection details are determined by the API Manager URL that you open, and the user ID with which you log in. If you are using the API Designer user interface, you provide the management server details and user ID in the login window that opens when you first launch API Designer; see [Logging into API Connect Designer](#).

- [Example - using multiple OAuth policies in an OAuth provider assembly](#)  
This example demonstrates the use of multiple OAuth policies in the assembly flow for a native OAuth provider.

## Related tasks

- [Creating a new REST OpenAPI definition](#)

## Related reference

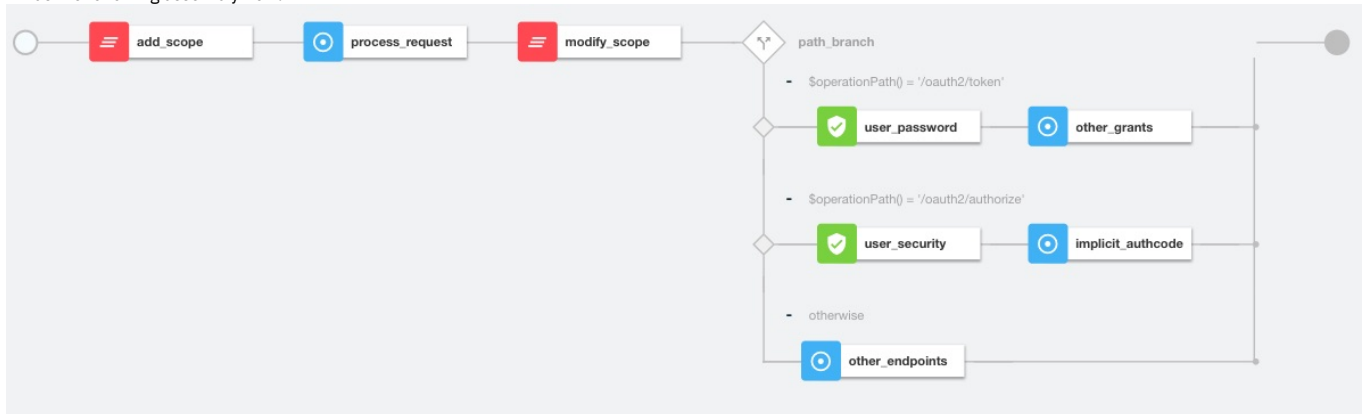
- [API Connect context variables](#)

## Example - using multiple OAuth policies in an OAuth provider assembly

This example demonstrates the use of multiple OAuth policies in the assembly flow for a native OAuth provider.

The example is based on the default assembly that is generated when you create a native OAuth provider, and is customized with the addition of `gatewayscript` policies that use [OAuth context variables](#) to manipulate the OAuth flow. For details on creating a native OAuth provider, see [Configuring a native OAuth provider](#).

It has the following assembly flow:



The following sections describe the OpenAPI source code that underlies each of the policies in the assembly; for the complete assembly code, download [multiple\\_oauth\\_policies.txt](#).

## Sample policy to add a custom scope

The `add_scope` policy is a `gatewayscript` policy that adds a custom scope to the request.

The underlying OpenAPI source YAML is as follows:

```
- gatewayscript:
 version: 2.0.0
 title: add_scope
 source: |-
 // Add another custom scope to the request
 let scope = context.get("request.parameters.scope.values[0]");
 if (scope)
 context.set("oauth.processing.scope", scope + " custom");
```

## Validate the initial OAuth request

The first `process_request` policy is an `oauth` policy that processes the initial request and verifies that the request is valid. The result of the processing is stored automatically in the [oauth.processing](#) context variables for use, as required, by the next OAuth policy in the assembly flow.

The underlying OpenAPI source YAML is as follows:

```
- oauth:
 title: process_request
 version: 2.0.0
 description: >-
 This oauth policy performs all OAuth/OpenID Connect protocol steps
```

```
that are needed for OAuth Validation by default. The inputs and
outputs of each of the steps are driven by documented context
variables. Add or remove the Supported OAuth Components as required.
oauth-provider-settings-ref:
 default: custom-form
supported-oauth-components:
 - OAuthValidateRequest
```

## Sample policy to modify the scope

---

The `modify_scope` policy is a `gatewayscript` policy that modifies the scope depending on the calling application.

The underlying OpenAPI source YAML is as follows:

```
- gatewayscript:
 version: 2.0.0
 title: modify_scope
 source: |-
 let admin_id = '1f1a2aa4-db9f-4423-b2f1-e2572b12123a';

 // Check application and modify the scope
 let app = context.get("oauth.processing.client_id");
 let scope = context.get("oauth.processing.scope");
 if (app === admin_id) {
 context.set("oauth.processing.scope", scope + " admin");
 } else {
 context.set("oauth.processing.scope", scope + " customer");
 }
 }
```

## Branch conditionally according to the OAuth path

---

The `path_branch` policy is a `switch` policy that branches according to the different OAuth paths to process the resource owner.

The underlying OpenAPI source YAML is as follows:

```
- switch:
 version: 2.0.0
 title: path_branch
 case:
 - condition: ($operationPath() = '/oauth2/token')
 execute:
 .
 .
 .
 definition of the user_security and other_grants policies
 .
 .
 .
 - condition: ($operationPath() = '/oauth2/authorize')
 execute:
 .
 .
 .
 definition of the user_password and implicit_authcode policies
 .
 .
 .
 - otherwise:
 execute:
 .
 .
 .
 definition of the other_endpoints policy
 .
 .
 .
```

## Process the user name and password, and enable grant type component

---

The following two policies operate on the token endpoint.

- The `user_password` policy for password grant type processes the user name and password from the `x-www-form-urlencoded` body. The authentication method is derived from the User Security settings in the OAuth provider; see [Configuring user security for a native OAuth provider](#).
- The `other_grants` policy is an `oauth` policy that enables the `OAuthGenerateAccessToken`, `OAuthVerifyAZCode`, `OAuthVerifyRefreshToken`, and `OAuthCollectMetadata` components to perform the operations for client credentials, authorization code, refresh token, and password grant types.

The underlying OpenAPI source YAML is as follows:

```
- user-security:
 title: user_password
 version: 2.0.0
 description: ''
 factor-id: default
 extract-identity-method: context-var
 user-context-var: request.parameters.username.values
 pass-context-var: request.parameters.password.values
 ei-stop-on-error: false
 user-auth-method: auth-url
 au-stop-on-error: false
 auth-url: 'http://httpbin.org/basic-auth/user/pass'
 user-az-method: authenticated
```

```

az-stop-on-error: true
auth-response-headers-pattern: (?)*x-api*
auth-response-header-credential: X-API-Authenticated-Credential
- oauth:
 title: other_grants
 version: 2.0.0
 description: >-
 This oauth policy performs all OAuth/OpenID Connect
 protocol steps that are needed for token path by default.
 The inputs and outputs of each of the steps are driven by
 documented context variables. Add or remove the Supported
 OAuth Components as required.
 oauth-provider-settings-ref:
 default: custom-form
 supported-oauth-components:
 - OAuthGenerateAccessToken
 - OAuthVerifyAZCode
 - OAuthVerifyRefreshToken
 - OAuthCollectMetadata

```

## Perform authorization checks, and enable grant type components

These following two policies operate on the authorize endpoint.

- The `user_security` policy configuration is derived from the User Security settings in the OAuth provider; see [Configuring user security for a native OAuth provider](#).
- The `implicit_authcode` policy is an `oauth` policy that enables the `OAuthGenerateAZCode`, `OAuthGenerateAccessToken`, and `OAuthCollectMetadata` components to perform the operations for the implicit and authorization code grant types.

The underlying OpenAPI source YAML is as follows:

```

- user-security:
 title: user_security
 version: 2.0.0
 description: >-
 This user security policy performs EI(basic) and AU(auth
 url) check for oauth assembly. Change the security check
 method as required
 factor-id: default
 extract-identity-method: basic
 ei-stop-on-error: true
 user-auth-method: auth-url
 au-stop-on-error: true
 user-az-method: authenticated
 az-stop-on-error: true
 auth-response-headers-pattern: (?)*x-api*
 auth-response-header-credential: X-API-Authenticated-Credential
 auth-url: 'http://httpbin.org/basic-auth/user/pass'
- oauth:
 title: implicit_authcode
 version: 2.0.0
 description: >-
 This oauth policy performs all OAuth/OpenID Connect
 protocol steps that are needed for az code path by
 default. The inputs and outputs of each of the steps are
 driven by documented context variables. Add or remove the
 Supported OAuth Components as required.
 oauth-provider-settings-ref:
 default: custom-form
 supported-oauth-components:
 - OAuthGenerateAZCode
 - OAuthGenerateAccessToken
 - OAuthCollectMetadata

```

## Process all other endpoints

The `otherwise` condition catches all other endpoints such as the introspect and revoke endpoints. The `other_endpoints` policy in the `otherwise` condition is an `oauth` policy that enables the `OAuthIntrospectToken`, and `OAuthRevokeTokencomponents` components to perform the operations for introspect and revoke.

The underlying OpenAPI source YAML is as follows:

```

- oauth:
 title: other_endpoints
 version: 2.0.0
 description: >-
 This oauth policy performs all OAuth/OpenID Connect
 protocol steps that are needed for all other paths by
 default. The inputs and outputs of each of the steps are
 driven by documented context variables. Add or remove the
 Supported OAuth Components as required.
 oauth-provider-settings-ref:
 default: custom-form
 supported-oauth-components:
 - OAuthIntrospectToken
 - OAuthRevokeToken

```

## Related concepts

- [API policies and logic constructs](#)

## Related reference

- [OAuth context variables](#)

## Related information

- [Configuring a native OAuth provider](#)

## Parse

Use the Parse policy to control the parsing of an input document. When the input document is a JSON string, the string is parsed instead of copied over.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0
	2.1.0 (DataPower API Gateway Version 10.0.3.0 or later)
	2.2.0 (DataPower API Gateway Version 10.5.0.5 or later)

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [parse](#).

## About Parse

With the parse policy, you can retrieve parse settings by using the following methods in increasing order of precedence.

1. Specify a valid parse settings configuration from which to retrieve its properties as the default parse settings.  
Note: If you change the default name, you must separately configure a corresponding ParseSettings object on the DataPower API Gateway.
2. Specify a literal string as serialized XML or JSON properties as parse settings. These properties take precedence over any existing default properties. The literal string must be XML or JSON formatted.
  - The XML format is "`<ParseSettings><propertyName>propertyValue</propertyName></ParseSettings>`".
  - The JSON format is "`{ \"ParseSettings\" : { \"propertyName\" : \"propertyValue\" } }`".
3. A URL that represents a named context from which to retrieve the serialized XML or JSON properties as parse settings. These properties take precedence over any existing literal or default properties.

## Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Parse policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is <b>parse</b> .	string
Description	No	A description of the policy.	string
Use Content Type	No	If this setting is enabled and the parse setting is configured to detect the document type, the parse operation uses the <b>Content-Type</b> specified in the request headers. If this setting is enabled and the document type in the parse setting is configured for either JSON or XML, the parse operation uses the <b>Content-Type</b> specified in the request headers and fails if the <b>Content-Type</b> in the request headers does not match the parse setting.  Enabling this setting is applicable only when the expected <b>Content-Type</b> is either JSON or XML.  If this setting is not enabled, the parse operation uses either the document type that is specified in the parse setting, or the detected document type if the parse setting is configured to detect the document type.  The check box is cleared by default.	boolean
Parse settings object reference	No	An existing valid object from which to retrieve default property values for the dynamic object.	string

Property label	Required	Description	Data type
Document type	No	<p>The type of document to parse. Select one of the following options:</p> <ul style="list-style-type: none"> <li>Detect: the document type is detected and the payload is parsed accordingly. This type is the default option.</li> <li>JSON: the payload is parsed as JSON.</li> <li>XML: the payload is parsed as XML.</li> <li>GraphQL: parsing is performed in one of the following ways: <ul style="list-style-type: none"> <li>As GraphQL in the body of a POST request. The GraphQL is in the following format: <pre> {   me {     name   } } </pre> </li> <li>As a JSON string in the body of a POST request that contains a GraphQL query. The JSON object is in the following format: <pre> {   "query": "query MyData (\$idVar: Int!) { user (id: \$idVar) { name }}",   "operationName": "MyData",   "variables": {"idVar": 123} } </pre> </li> <li>Through the URL query of a GET request. The URL is in the following format: <pre> https://myapi/graphql?query={me{name}} </pre> <p>The following example includes the <b>variables</b> and <b>operationName</b> properties, with URL encoding (line-breaks are for table formatting only, your value will not include line-breaks):</p> <pre> https://hostname/basepath/graphql?query=query%20fetchAccounts%20(\$limit:%20Int)%20{accounts(limit:%20\$limit)%20{name{first,last}}&amp;variables={"limit":100}&amp;operationName=fetchAccounts </pre> </li> </ul> </li> </ul> <p>Query variables in a GET request can be sent as a JSON string in a query parameter called <b>variables</b>. If the query contains several named operations, an <b>operationName</b> query parameter is required to determine which operation is to be executed.</p>	string
Maximum document size	No	<p>Optionally, the maximum document size in bytes of a JSON, XML, or GraphQL input document that the parse operation accepts. This setting provides threat protection by enforcing the maximum document size. Enter a value in the range 0 - 5368709121. A document is rejected when its size exceeds the maximum size. A value of 0 indicates an unlimited document size. When set to 0, the operation result does not return the document size. The default value is 4194304.</p>	integer
Maximum nesting depth	No	<p>Optionally, the maximum nesting depth of an XML, JSON, or GraphQL message that the parse operation accepts. This setting provides threat protection by enforcing limits in the message.</p> <ul style="list-style-type: none"> <li>When XML, it is the maximum level of element depth.</li> <li>When JSON, it is the maximum level of nested label-value pairs, the maximum number of nested arrays, or the maximum number of combination of label-value pairs and arrays.</li> <li>When GraphQL, it is the maximum level of nested selection sets.</li> </ul> <p>Enter a value in the range 0 - 4096. A document is rejected when its nesting depth exceeds the maximum depth. A value of 0 indicates unlimited nesting depth. When set to 0, the operation result does not return the nesting depth.</p> <p>The default value is 512.</p>	integer
Maximum width	No	<p>Optionally, the maximum width of the payload that the parse operation accepts.</p> <ul style="list-style-type: none"> <li>When the input document is XML, this property specifies the maximum number of attributes for an element or the maximum number of child elements for an element.</li> <li>When the input document is JSON, this property specifies the maximum number of properties on a JSON object or the maximum number of JSON items in a JSON array.</li> <li>When the input document is GraphQL, this property specifies the maximum number of selections in a selection set.</li> </ul> <p>Enter a value in the range 0 - 65535. The document is rejected when its width exceeds the maximum width. A value of 0 indicates unlimited width. When set to 0, the operation result does not return the width of the document.</p> <p>The default value is 4096.</p>	integer
Maximum name length	No	<p>Optionally, the maximum name length in bytes in a document that the parse operation accepts.</p> <ul style="list-style-type: none"> <li>For XML, it is the length of the name portion of a tag.</li> <li>For JSON, it is the length of the label portion of the JSON label-value pair.</li> <li>For GraphQL, it is the maximum length of the identifiers, including field names and directive names.</li> </ul> <p>The length includes any white space that is contained between tags in XML or quotation marks in JSON. Enter a value in the range 0 - 8192. A document is rejected when its name length exceeds the maximum length. A value of 0 indicates unlimited name length. When set to 0, the operation result does not return the name length of a document.</p> <p>The default value is 256.</p>	integer



Property label	Required	Description	Data type
Maximum value length	No	<p>Optionally, the maximum value length in bytes in a document that the parse operation accepts.</p> <ul style="list-style-type: none"> <li>For XML, it is the length of an attribute or the length of a text value.</li> <li>For JSON and GraphQL, it is the length of a string value.</li> </ul> <p>The length includes any white space that is contained between tags in XML or quotation marks in JSON. Enter a value in the range 0 - 5368709121. A document is rejected when its value length exceeds the maximum length. A value of 0 indicates unlimited value length. When set to 0, the operation result does not return the value length of a document.</p> <p>The default value is 8192.</p>	integer
Maximum number of unique names	No	<p>Optionally, the maximum number of unique names in a JSON or XML document that the parse operation accepts.</p> <ul style="list-style-type: none"> <li>For XML, it is the number of unique XML local names.</li> <li>For JSON, it is the number of unique JSON labels.</li> </ul> <p>Enter a value in the range 0 - 1048575. A document is rejected when the number of unique names in the document exceeds the maximum number. A value of 0 indicates an unlimited number of unique names. When set to 0, the operation result does not return the number of unique names in a document.</p> <p>The default value is 1024.</p>	integer
Maximum number of unique prefixes	No	<p>Optionally, the maximum number of unique XML namespace prefixes in a document that the parse operation accepts. This limit counts multiple prefixes defined for the same namespace, but does not count multiple namespaces defined in different parts of the input document under a single prefix. Enter a value in the range 0 - 262143. A document is rejected when the number of unique prefixes in the document exceeds the maximum number. A value of 0 indicates an unlimited number of unique prefixes. When set to 0, the operation result does not return the number of unique prefixes in a document.</p> <p>The default value is 1024.</p>	integer
Maximum number of unique namespaces	No	<p>Optionally, the maximum number of unique XML namespace URIs that the parse operation accepts. This limit counts all XML namespaces, regardless of how many prefixes are used to declare them. Enter a value in the range 0 - 65535. A document is rejected when the number of unique namespaces in the document exceeds the maximum number. A value of 0 indicates an unlimited number of unique namespaces. When set to 0, the operation result does not return the number of unique namespaces in a document.</p> <p>The default value is 1024.</p>	integer
Maximum number length	No	<p>Optionally, the maximum number of bytes in the value portion of a JSON label-value pair when the value is a number. The number must be a contiguous string of bytes that contain no white space. The number can include a minus sign and a positive or negative exponent. Enter a value in the range 0 - 256. A document is rejected when the number length in the document exceeds the maximum length. A value of 0 indicates unlimited number length. When set to 0, the operation result does not return the number length in a document.</p> <p>The default value is 128.</p>	integer
Parse settings literal configuration	No	A literal string as serialized XML or JSON properties that are merged into the dynamic object. These properties take preference over any existing default properties.	string
Parse settings URL reference	No	A URL that represents a named context from which to retrieve the serialized XML or JSON properties that are merged into the dynamic object. These properties take preference over any existing literal or default properties.	string
Message for parsing	No	The name of a variable in the API context. The content of the <b>body</b> field of the variable is the input to the policy. The default variable name is <b>message</b> .	string
Message for resulting parsed data	No	The name of a variable in the API context. The content of the body field of the variable is the output of the parse operation. The parse metrics of the parsed document can be stored in a different part of the message. The default variable name is the same as the input name, so by default the input message is overwritten by the output message.	string

Use the YAML source to set the following option.

Property	Required	Description	Data type
strict-utf8-encoding	No	<p>For JSON documents, whether to enforce strict UTF-8 encoding throughout the entire document.</p> <ul style="list-style-type: none"> <li>When <b>on</b>, the entire document is checked for valid UTF-8 encoding.</li> <li>When <b>off</b>, only the first few bytes are checked for proper encoding. The rest of the document is assumed to be in the same encoding.</li> </ul> <p>The default value is <b>off</b>.</p>	boolean

For examples, see [parse](#).

## Proxy

Apply the Proxy policy to invoke another API within your assembly, particularly if the separate API contains a large payload. The response from the backend is stored in the **message.body** and in the response object variable if it is defined. Only one policy is permitted to be run per unique assembly flow.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway, functionality provided by <a href="#">Invoke</a>	

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [proxy](#).

## About

Only one Proxy policy is permitted to be run per unique flow of your assembly. More than one Proxy policy can be applied, if they are contained in mutually exclusive branches of the assembly.

You can use the Proxy policy to return multipart form data, that is, when the response is set to `Content-Type: multipart/related`. However, Proxy must be the last policy in the assembly, otherwise the response that is received can be manipulated during subsequent steps, thereby causing the multipart form data to be lost.

The proxy policy, if inside a conditional policy, must be the **final** policy to be executed in the API. If you need further processing afterward, use the [invoke](#) policy rather than the proxy policy.

## Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Proxy policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is <b>proxy</b> .	string
Description	No	A description of the policy.	string
Invoke URL	Yes	Specifies a URL for the target service. For a SOAP API, a URL is added by default. Where possible, the Proxy URL value is pre-supplied from information that is defined in the imported WSDL.	string
TLS profile	No	Specifies a TLS profile to use for the secure transmission of data.	string
Timeout	Yes	The time to wait before a reply back from the endpoint (in seconds). The default value is <b>60</b> .	integer
Username	No	The username to use for HTTP Basic authentication.	string
Password	No	The password to use for HTTP Basic authentication.	string
HTTP Method	Yes	The HTTP method to use for the proxy. Valid values are: <ul style="list-style-type: none"> <li>• Keep</li> <li>• GET</li> <li>• POST</li> <li>• PUT</li> <li>• DELETE</li> <li>• PATCH</li> <li>• HEAD</li> <li>• OPTIONS</li> </ul> The default value is <b>Keep</b> . By using <b>Keep</b> , or removing the property from the source, the HTTP method from the incoming request is used.	string
Compression	No	Select this check box to enable Content-Encoding compression on upload. The check box is cleared by default.	boolean

Property label	Required	Description	Data type
Cache Type	No	<p>The cache type determines whether to cache documents, honoring or overriding the HTTP Cache Control directives received in the response from the target URL. This property takes effect only when a response is received, otherwise the policy always returns the non-expired response that was previously saved in cache. Valid values are:</p> <p><b>Protocol</b> The cache behavior is determined by the Cache-Control headers on the response, in accordance with RFC 7234. To optimize performance, if the gateway receives more than one request for a resource that is not in the cache but could be cached when the response from the target URL is received, the gateway sends only one request to the target URL; the remaining requests are not processed until the response from the first request has been received and the cache behavior has been determined from this response. If the response indicates that caching is possible, the gateway responds to all waiting requests with the cached resource. If the response indicates that caching is not possible, the gateway sends all waiting requests to the target URL.</p> <p>Use this option only if you expect that responses from the target URL can be cached, in which case it should improve performance and limit the demand on the target URL. If, however, the target URL never indicates that the gateway should cache its response, performance might be impaired when compared to the No Cache option.</p> <p><b>No Cache</b> Responses from the target URL are not cached on the gateway regardless of any caching headers returned. In this case, every request from the client is sent to the target URL. Use this option if you do not want to cache any of the backend responses on the gateway, or if it is unlikely that a response from the target URL will allow caching through the Cache-Control header settings.</p> <p><b>Time to Live</b> This option is similar to the Protocol option except it allows you to specify the amount of time that you want the successful response from the invoke or proxy to remain in the cache. Use this option only if you expect that responses from the target URL can be cached.</p> <p>The default value is Protocol.</p>	string
Time to Live	No	Specifies the amount of time in seconds that the response stays in the cache. Applies only if the property Cache type is set to <b>Time to Live</b> . Enter a value in the range 5 - 31708800. The default value is 900.	integer
Cache key	No	Specifies the unique identifier of the document cache entry. If omitted, the entire URL string is used as the key.	string
Stop on error	No	Select the errors that, if thrown during the policy execution, cause the assembly flow to stop. If there is a <b>catch</b> flow configured for the error, it is triggered to handle the error thrown. If an error is thrown and there are no errors selected for the Stop on error setting, or if the error thrown is not one of the selected errors, the policy execution is allowed to complete, and the assembly flow continues.	string
Response object variable	No	The name of a variable that will be used to store the response data from the request. This variable can then be referenced in other actions, such as 'Map'.	string
<b>X-Forwarded</b> header	No	<p>This header can be provided by</p> <ol style="list-style-type: none"> <li>If <b>X-Forwarded-Host</b> exists, processing continues. If it does not exist prior to calling the proxy policy, it is set with the value of the <b>Host</b> header.</li> <li>The <b>X-Forwarded-For</b> header is always set, in all cases. This header maintains breadcrumbs, showing a comma-separated list of IPs, from the client through any preceding proxy.</li> <li>If all three of <b>X-Forwarded-Host</b>, <b>X-Forwarded-Port</b>, and <b>X-Forwarded-Proto</b> headers are missing at the time of calling the proxy policy, they are set automatically. To prevent this, set <b>X-Forwarded-Host</b> header to some value before calling the proxy policy.</li> </ol>	string

## Related tasks

- [Creating a new REST OpenAPI definition](#)

## Related information

- [TLS profiles](#)



## Rate Limit

Use the Rate Limit policy to apply one or more rate, burst, or count limits at any point in your API assembly flow. Rate and burst limits restrict the number of calls made to an API in a specified time period, while count limits impose a strict limit on the total number of calls.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0
	2.1.0 (DataPower API Gateway Version 10.0.1.0 or later)
	2.2.0 (DataPower API Gateway Version 10.0.2.0 or later)
	2.3.0 (DataPower API Gateway Version 10.5.0.0)

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [ratelimit](#).

## About

The defined rate, burst, and count limits are applied to whatever follows in the assembly flow. For example, if a Rate Limit policy is placed before an Invoke policy, and the call made by the Invoke policy exceeds the limits defined by the Rate Limit policy, the API call itself fails.

Note: For information about rate limits and burst limits in API Connect, see [Understanding rate limits for APIs and Plans](#).

## Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Rate Limit policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is <code>ratelimit</code> .	string
Description	No	A description of the policy.	string
Source	Yes	The location of all the rate limit, burst limit, and count limit definitions that are included in this policy. Select one of the following options: <ul style="list-style-type: none"> <li>Catalog by Name: the limits to be applied are defined in the appropriate <code>api-collection</code> object on the DataPower API Gateway, which is the object that represents your API Connect Catalog in the gateway configuration. For details of how to configure a rate, burst, or count limit in the <code>api-collection</code> object, see <a href="#">Configuring a rate, burst, or count limit on the DataPower API Gateway</a>.</li> <li>Plan by Name: the limits to be applied are assembly burst limits or assembly count limits defined on the Plan to which the calling application is subscribed. For details on how to configure assembly burst limits and assembly count limits in a Plan, see <a href="#">Editing a draft Product</a>. Note: The following conditions apply to a Plan by Name rate limit policy: <ul style="list-style-type: none"> <li>Every Plan that contains the API that is being limited must have the named rate limit applied.</li> <li>A Plan by Name rate limit policy cannot be used with automatically generated Products.</li> </ul> </li> <li>Gateway by Name: the limits to be applied are defined in the <code>apigw</code> object, named <code>apiconnect</code>, on the DataPower API Gateway gateway. For details of how to configure a rate, burst, or count limit in the <code>apigw</code> object, see <a href="#">Configuring a rate, burst, or count limit on the DataPower API Gateway</a>.</li> <li>Plan Default: the rate and burst limits that are applied are the default ones configured in the Plan to which the calling application is subscribed. For details on how to configure default Plan rate and burst limits, see <a href="#">Editing a draft Product</a>. Note: The use of this option does not disable any automatic preflow rate limit policy that might be in place, and could result in the application of the rate limiting being repeated. For more information on preflow policies, see <a href="#">Customizing the preflow policies</a>.</li> </ul>	string
Rate Limit Name	Yes <sup>1</sup>	If Source property is set to Catalog by Name, the name of a rate limit as defined in the DataPower API Gateway configuration, To display the Rate Limit Name field, click Add Rate Limit; you can add as many rate limits as you want.	string

<sup>1</sup>You must provide at least one rate, burst, or count limit.

Property label	Required	Description	Data type
Burst Limit Name	Yes <sup>2</sup>	<p>If Source property is set to Catalog by Name, the name of a burst limit as defined in the DataPower API Gateway configuration.</p> <p>If Source is set to Plan by Name, the name of an assembly burst limit in a Plan.</p> <p>To display the Burst Limit Name field, click Add Burst Limit; you can add as many burst limits as you want.</p> <p><sup>2</sup>You must provide at least one rate, burst, or count limit.</p>	string
Count Limit Name	Yes <sup>3</sup>	<p>If Source property is set to Catalog by Name, the name of a count limit as defined in the DataPower API Gateway configuration.</p> <p>If Source is set to Plan by Name, the name of an assembly count limit in a Plan.</p> <p>To display the Count Limit Name field, click Add Count Limit; you can add as many count limits as you want.</p> <p><sup>3</sup>You must provide at least one rate, burst, or count limit.</p>	string
Operation	No	<p>For a rate limit, select one of the following options:</p> <ul style="list-style-type: none"> <li>Consume: the policy reduces the balance of available requests that remain in the interval defined by the rate limit specified in the Rate Limit Name field. The amount by which the remaining balance is reduced is the value that results from the weight expression defined in the specified rate limit.</li> <li>Replenish: the policy increases the balance of available requests that remain in the interval defined by the rate limit specified in the Rate Limit Name field. The amount by which the remaining balance is increased is the value that results from the weight expression defined in the specified rate limit.</li> </ul> <p>The purpose of the replenish operation is to restore an appropriate amount of remaining balance in cases where the amount calculated by an earlier consume operation turns out to be too high. For example, for an assembly rate limit with a limit of 100, a consume operation with a weight of 60 uses 60 requests, leaving 40 requests remaining for the interval. If the weight expression in the subsequent replenish operation computes to 15, meaning that the weight expression calculated that the actual request usage was only 45, the replenish operation increases the remaining requests by 15, to a remaining balance of 55.</p> <p>Note that the replenish operation cannot increase the remaining requests allowed beyond the original limit defined in the assembly rate limit scheme. In this example, a replenish operation cannot replenish requests beyond 100. A replenish operation is skipped if the weight expression evaluates to a value of less than 1.</p> <p>The default value is Consume.</p> <p>For a count limit, select one of the following options:</p> <ul style="list-style-type: none"> <li>Increment: the current count is incremented by the value that results from the weight expression defined in the count limit specified in the Count Limit Name field.</li> <li>Decrement: the current count is decremented by the value that results from the weight expression defined in the count limit specified in the Count Limit Name field.</li> </ul> <p>By default, the count limit is automatically decremented at the end of the API assembly by the total amount incremented during the assembly, minus any decrement operations that are defined in the assembly. You do not need to explicitly define decrement operations unless you want the decrements to occur at specific points in the assembly. However, if the auto decrement feature has been explicitly disabled in the count limit configuration, you must include a corresponding decrement operation, otherwise when the count limit is reached all future calls are blocked. For more information on configuring a count limit, see <a href="#">Configuring a rate, burst, or count limit on the DataPower API Gateway</a>.</p> <p>The default value is Increment.</p> <p>For a burst limit, the only option is Consume, meaning that the limit applied is that defined in the burst limit specified in the Burst Limit Name field.</p>	string

- [Configuring a rate, burst, or count limit on the DataPower API Gateway](#)

If you want to include a rate limiting policy in your API assembly flow, you must first configure the required rate, burst, and count limits on the Gateway. Rate and burst limits restrict the number of calls that an application can make to API in specified time period, while count limits impose a strict limit on the total number of calls.



## Configuring a rate, burst, or count limit on the DataPower API Gateway

If you want to include a rate limiting policy in your API assembly flow, you must first configure the required rate, burst, and count limits on the Gateway. Rate and burst limits restrict the number of calls that an application can make to API in specified time period, while count limits impose a strict limit on the total number of calls.

### About this task

You configure rate, burst, and count limits on the DataPower® API Gateway by defining them in one or more configuration files that you package and then add to the Gateway as a Gateway extension.

You define *Catalog scoped* rate, burst, and count limits in the appropriate **api-collection** object on the DataPower API Gateway, which is the object that represents your API Connect Catalog in the gateway configuration. The defined rate, burst, and count limits will be available to be included in [Rate Limit](#) policies in the assembly flows of all the APIs that are published to that Catalog.

You define *Gateway scoped* rate, burst, and count limits in the appropriate **apigw** object on the DataPower API Gateway. The defined rate, burst, and count limits will be available to be included in [Rate Limit](#) policies in the assembly flows of all the APIs that are published to that Gateway.

Note:

- Only one Gateway extension can be in use at any one time, so you should package all required rate, burst, and count limits in a single Gateway extension.
- To apply a Catalog scoped or Gateway scoped rate, burst, or count limit, it must be referenced in a [Rate Limit](#) policy in an API assembly.
- You should ensure that all rate, burst, and count limit names are unique across all your Plans and Catalogs. When an API is called, information about any rate, burst, or count limits (such as name, limiting criteria, and remaining number of calls) is returned in the HTTP headers of the response. If two or more limit definitions with exactly the same name are applied during the invocation of an API, information about only **one** of them is returned. However, **all** limiting criteria are applied.

## Procedure

---

1. Create a `.cfg` file that includes the DataPower API Gateway CLI commands that define the rate, burst, or count limit.

To define a rate limit, use the **assembly-rate-limit** command. To define a burst limit, use the **assembly-burst-limit** command. To define a count limit, use the **assembly-count-limit** command. For details of the command parameters, see [assembly-rate-limit](#), [assembly-burst-limit](#), and [assembly-count-limit](#) in the DataPower product documentation.

Example 1: Configuring a Catalog scoped rate limit

```
top; config;
switch apiconnect;

api-collection myorg_mycatalog_collection
 assembly-rate-limit 30perlmin 30 1 minute on off on on off off "" "my.vars.expected-amount - my.vars.actual-amount"
exit
```

In this example, the **assembly-rate-limit** command specifies a rate limit name, and the API call limits that you want to impose, 30 calls per minute in this case. The final, optional argument, `"my.vars.expected-amount - my.vars.actual-amount"`, is a weight expression. You use weight expressions in cases where you want to restore an appropriate amount of remaining rate limit balance when an earlier usage calculation in your assembly flow turns out to be too high. You define two rate limiting policies in your assembly flow, one with a **consume** operation, and a subsequent one with a **replenish** operation that restores some of the remaining rate limit balance if appropriate. The weight expression calculates the amounts to be consumed or replenished; for more information, see [Rate Limit](#).

The **api-collection** command specifies the **api-collection** object in which you want to configure the rate, burst, or count limits. The name of the **api-collection** object has the following format:

```
org_catalog_collection
```

where:

- `org` is the name of the provider organization that contains your Catalog.
- `catalog` is the name of the Catalog.

This **api-collection** object name is assigned automatically by the Gateway the first time that a Product is published to the Catalog.

Example 2: Configuring a Catalog scoped count limit

```
top; config;
switch apiconnect;

api-collection myorg_mycatalog_collection
 assembly-count-limit maxsixty 60 1 on on off off off off "" "1" off
exit
```

The final argument is the **autodecrement** property that controls whether the count limit is automatically decremented at the end of an API assembly. The default value is **on**, meaning that at the end of the assembly the count limit is automatically decremented by the total amount incremented during the assembly, minus any decrement operations that are defined in the assembly. You do not need to explicitly define decrement operations unless you want the decrements to occur at specific points in the assembly.

Example 3: Configuring a Gateway scoped rate limit

```
top; config;
switch apiconnect;

apigw apigw_object_name
 assembly-rate-limit 30perlmin 30 1 minute on off on on off off "" "my.vars.expected-amount - my.vars.actual-amount"
exit
```

The **apigw** command specifies the name of the **apigw** object in which you want to configure the rate, burst, or count limits; the **apigw** object name is the same as the name of the API Connect domain in your DataPower API Gateway configuration.

2. Package the `.cfg` file in a `.zip` file.  
You can package multiple `.cfg` files in a single `.zip` file if you want; the `.cfg` files must all be at the root of the `.zip` file.
3. Add the `.zip` file to the DataPower API Gateway as a Gateway extension. For details, see [Configuring your Gateway server extensions](#).

## What to do next

---

Apply the rate, burst, or count limits to an API by adding a [Rate Limit](#) policy to the API assembly flow.

## Related information

---

- [Gateway extension guidelines - DataPower API Gateway.](#)



## Redaction - DataPower API Gateway

Use the Redaction policy to completely remove or to redact specified fields from the Request body, the Response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.

### Gateway support

Note: This page describes the Redaction policy implementation in the DataPower® API Gateway. If you are using the DataPower Gateway (v5 compatible), see [Redaction - DataPower Gateway \(v5 compatible\)](#).

For information on the different types of gateway, see [API Connect gateway types](#).

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [redact - DataPower API Gateway](#).

Note: With the DataPower API Gateway, the input to the Redaction policy must be parsed data. One way to produce parsed data is to use a [Parse](#) policy before a Redaction policy in your assembly flow, which provides explicit control of the parse action.

### Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Redaction policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is <b>redact</b> .	string
Description	No	A description of the policy.	string
Root	No	Specifies the data source that contains the content to redact or remove. If no value is entered in the Root field, the action is applied to the entire API context. You can use any supported JSONata path expression.  If you want to apply the action to either request or response data, specify a value of <code>message.body</code> . The actual content to which the action is applied then depends on the positioning of the Redaction policy in the overall assembly flow; for example: <ul style="list-style-type: none"> <li>• If positioned at the beginning, the action is applied to the client request.</li> <li>• If positioned after an Invoke policy, the action is applied to the response from the back end.</li> <li>• If positioned at the end, the action is applied to the response that is returned to the client.</li> </ul> If, in your assembly flow, the Redaction policy is used after a <a href="#">Log</a> policy that specifies Gather-only mode, specify a Root value of <code>log.request_body</code> for the logged request payload, or <code>log.response_body</code> for the logged response payload.	string
Path	Yes	Specifies a JSONata path expression that identifies the content to redact or remove from the source. For more information, see <a href="#">Constructing JSONata expressions to redact fields</a>	string
Action	No	Specifies whether you want to remove or redact the content. Choose one of the following options: <ul style="list-style-type: none"> <li>• Remove: Completely removes the specified fields.</li> <li>• Redact: Redacts (obfuscates with "*"s) the fields to block out the data.</li> </ul> The default value is Redact.  Note: If a numerical value is being redacted, the redacted value is depicted as <b>*****</b> and the type is changed to <b>string</b> .	string

Tip: You can optionally click Add action to specify JSONata expressions for additional fields that you want to remove or redact from the specified content source.

- [Constructing JSONata expressions to redact fields](#)

To define a field for redaction or removal when using the Redaction policy with the DataPower API Gateway, you supply a JSONata expression that defines the path to the field that you want to redact or remove.



## Constructing JSONata expressions to redact fields

To define a field for redaction or removal when using the Redaction policy with the DataPower® API Gateway, you supply a JSONata expression that defines the path to the field that you want to redact or remove.

Note: For a list of known limitations to DataPower API Gateway support for JSONata, see [Limitations to support for JSONata](#).  
The following JSONata functions are supported:

- Aggregation functions:
  - `$average(array)`
  - `$max(array)`
  - `$min(array)`
  - `$sum(array)`
- Array functions:
  - `$append(array1, array2)`
  - `$count(array)`
  - `$reverse(array)`
  - `$sort(array [, function])`
  - `$zip(array1, ...)`
- Boolean functions:
  - `$boolean(arg)`
  - `$exists(arg)`
  - `$not(arg)`
- Numeric functions:
  - `$abs(number)`
  - `$ceil(number)`
  - `$floor(number)`
  - `$formatBase(number [, radix])`
  - `$number(arg)`
  - `$power(base, exponent)`
  - `$round(number [, precision])`
  - `$sqrt(number)`
- Object functions:
  - `$keys(object)`
  - `$lookup(object, key)`
  - `$merge(array<object>)`
  - `$spread(object)`
  - `$type(value)`

In addition to the standard values returned by `$type()`, the API gateway implementation of `$type()` can return the following values: `binary`, `empty`, `graphql`, `stream`, or `xml`.
- String functions:
  - `$contains(str, pattern)`
  - `$join(array[, separator])`
  - `$length(str)`
  - `$lowercase(str)`
  - `$match(str, pattern [, limit])`
  - `$pad(str, width [, char])`
  - `$replace(str, pattern, replacement [, limit])`
  - `$split(str, separator [, limit])`
  - `$string(arg)`
  - `$substring(str, start[, length])`
  - `$substringAfter(str, chars)`
  - `$substringBefore(str, chars)`
  - `$trim(str)`
  - `$uppercase(str)`
- You can use the following functional extensions to standard JSONata notation. Each extension corresponds to a part of the API context.

Table 1. Functional extensions to JSONata

Extension	Variable	Description
<code>\$header(name)</code>	<code>message.headers.name</code>	Message header
<code>\$httpVerb()</code>	<code>request.verb</code>	HTTP method of the request
<code>\$operationID()</code>	<code>api.operation.id</code>	ID of the operation
<code>\$operationPath()</code>	<code>api.operation.path</code>	Path of the operation
<code>\$queryParameter('name')</code>	<ul style="list-style-type: none"> <li>◦ <code>request.parameters.name.locations</code> The supported keyword is <code>query</code>.</li> <li>◦ <code>request.parameters.name.values</code></li> </ul>	Searches for the index of <code>query</code> in <code>request.parameters.name.locations</code> and returns <code>request.parameters.name.values[index]</code> , where <code>[index]</code> is the value for <code>query</code> in locations. Parameter values are not URL decoded.
<code>\$statusCode()</code>	<code>message.status.code</code>	Status code
<code>\$storageType([arg])</code>	<code>variable.body</code> You can specify any variable in the API context. When no variable is specified, the default variable <code>message.body</code> is used.	Storage type of the message. The supported values are <code>binary</code> , <code>empty</code> , <code>graphql</code> , <code>json</code> , <code>stream</code> , or <code>xml</code> .



Extension	Variable	Description
<code>\$urlParameter('name')</code>	<ul style="list-style-type: none"> <li><code>request.parameters.name.locations</code> The supported keywords are <code>path</code> and <code>query</code></li> <li><code>request.parameters.name.values</code></li> </ul>	<p>Searches for the index of <code>path</code> and <code>query</code> in <code>request.parameters.name.locations</code> and returns a single array that contains both <code>path</code> and <code>query</code> values from <code>request.parameters.name.values</code>. When the URL contains both path and query parameter values, the array includes the path values first followed by the query values. The values of each parameter type are added in the order that they are received. Parameter values are URL decoded.</p> <p>For example, the following URL contains both path and query parameter values.</p> <pre>http://example.com/petstore/cats/adopt?breed=Sphynx&amp;breed=Siamese</pre> <p>The <code>\$urlParameter('breed')</code> URL returns the following array of values.</p> <pre>[cats, adopt, Sphynx, Siamese]</pre> <p>In this example, the URL includes an API path that is configured as <code>/petstore/{breed}/{breed}</code>, where <code>breed</code> is configured to be a path parameter of the API path. As a result, <code>cats</code> and <code>adopt</code> are included in the output.</p>
<code>\$xpath(path, xpathExpression)</code>	You can specify any writable variable in the API context. The <code>xpathExpression</code> must be a literal string.	<p>Allows use of XPath expressions. The following example specifies all price elements in the source.</p> <pre>\$xpath(\$, '//price')</pre>

Table 2. Functional extensions to JSONata for GraphQL

Extension	Variable	Description
<code>\$gqlActiveOperation([graphql_message])</code>	<code>message.body</code>	Gets the active operation found in the specified GraphQL message. The <code>operationName</code> must be the same as the name of the active operation.
<code>\$gqlAlias(graphql_field_node)</code>	<code>message.body</code>	Gets the alias of a GraphQL field node.
<code>\$gqlFragments([graphql_message])</code>	<code>message.body</code>	Gets the fragments found in the specified GraphQL message.
<code>\$gqlName([graphql_node])</code>	<code>message.body</code>	Gets the node name. For operations, the node name is the <code>operationName</code> . For fields, fragment definitions, arguments, and other elements, the node name is the name of the element. By default, the <code>operationName</code> of <code>message.body</code> is retrieved.
<code>\$gqlOperations([graphql_message])</code>	<code>message.body</code>	Gets the operations found in the specified GraphQL message.
<code>\$gqlType([graphql_node])</code>	<code>message.body</code>	The operation type of the active operation is retrieved for Query, Mutation, and Subscription query types.

- You can use the `&` (concatenation) navigation operator.

You can use the following JSONata numeric operators:

- `+` (addition)
- `-` (subtraction)
- `*` (multiplication)
- `/` (division)
- `%` (modulo)

You can use the following JSONata comparison operators for number values or strings:

- `=`
- `!=`
- `<`
- `>`
- `<=`
- `>=`

You can also use the following operators and expressions.

- Parentheses to convert a sequence into an array, specify operator precedence, or compute complex expressions on a context value.
- Array ranges and predicate expressions.
- Single asterisk (`*`) and double asterisk (`**`) wildcard characters.

The following elements of a GraphQL query can be exposed in JSONata notation using the syntax shown.

- `query`  
The entire GraphQL query including operations and fragments.
- `operationName`  
For an anonymous operation, `operationName` can be empty.
- `~fragmentSpreadName`
- `on~typeCondition`
- `~~fragmentDefinitionName`

You can define the path in either of the following ways:

- [Use a JSONata expression](#)
- [Use the `\$xpath\(\)` JSONata extension](#)

## Using a JSONata expression

The path specified by the JSONata expression is relative to any value specified for the `root` property of the Redaction policy. If the `root` property has no value or is absent, begin the expression with the absolute content path. If the `root` property has a value then you can either begin the expression with `$` to use the `root` path directly, or you can provide a sub-path relative to the `root` path.

JSONata expressions can be used with content that is in either JSON or XML format.

#### Example 1

If the `root` property of the Redaction policy has no value or is absent, use the following expression to redact or remove all occurrences of the `price` field in the request and response data:

```
message.body.**.price
```

#### Example 2

If the `root` property of the Redaction policy has the value `log.request_body`, use the following expression to redact or remove all occurrences of the `price` field, specifically within an `item` element, in the logged request payload:

```
$.item.price
```

#### Example 3

If the `root` property of the Redaction policy has the value `log`, use the following expression to redact or remove all occurrences of the `price` field in the logged response payload:

```
response_body.**.price
```

The `**` descendant wildcard traverses all descendants at all hierarchical levels.

## Using the `$xpath()` JSONata extension

The `$xpath()` function has the following format:

```
$xpath(content_path, xpath_expression)
```

where:

- `content_path` is the path to the content that contains the field that you want to redact or remove.
- `xpath_expression` is the XPath expression that defines the field that you want to redact or remove.

The `content_path` is relative to any value specified for the `root` property of the Redaction policy. If the `root` property has no value or is absent, provide the absolute content path. If the `root` property has a value then you can either provide the value `$` for the `content_path` parameter to use the `root` path directly, or you can provide a sub-path relative to the `root` path.

The `$xpath()` function can be used only with content that is in XML format.

#### Example 1

If the `root` property of the Redaction policy has no value or is absent, use the following expression to redact or remove all occurrences of the `price` field in the request and response data:

```
$xpath(message.body, '//price')
```

#### Example 2

If the `root` property of the Redaction policy has the value `log.request_body`, use the following expression to redact or remove all occurrences of the `price` field, specifically within an `item` element, in the logged request payload:

```
$xpath($, 'item/price')
```

#### Example 3

If the `root` property of the Redaction policy has the value `log`, use the following expression to redact or remove all occurrences of the `price` field in the logged response payload:

```
$xpath(response_body, '//price')
```

The `//` expression in the second parameter selects all occurrences anywhere in the content.

DataPower Gateway (v5 compatible)

## Redaction - DataPower Gateway (v5 compatible)

Use the Redaction policy to completely remove or to redact specified fields from the Request body, the Response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.

## Gateway support

Note: This topic describes the Redaction policy implementation in the DataPower® Gateway (v5 compatible). If you are using the DataPower API Gateway, see [Redaction - DataPower API Gateway](#).

For information on the different types of gateway, see [API Connect gateway types](#).

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower Gateway (v5 compatible)	1.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [redact - DataPower Gateway \(v5 compatible\)](#).

## About

## Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Redaction policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is <b>redact</b> .	string
Description	No	A description of the policy.	string
Path	Yes	Specifies an XPath expression that defines the field to remove or redact. You can construct an XPath expression that is based on JSON or XML depending on whether your API requests and responses use a JSON or an XML format. If the payload is JSON, use the DataPower XML representation of the JSON content (JSONx) to construct the expression.  Note: Use a JSONx representation only to identify the XPath expressions for the fields to remove or redact. Do not change the format of any response bodies in API Manager. To learn more about constructing XPath expressions that are based on JSON or XML, see <a href="#">Constructing XPath expressions to redact fields</a> .	string
Action	Yes	Specifies whether you want to remove or redact the field. Valid values: <ul style="list-style-type: none"><li><b>remove</b>: Completely removes the specified field.</li><li><b>redact</b>: Redacts (obfuscates with "*"s) the field to block out the data.</li></ul> The default value is <b>redact</b> .  Note: If a numerical value is being redacted, the redacted value is depicted as ***** and the type is changed to <b>string</b> .	string
From	Yes	Specifies where to remove or redact the specified field from. Valid values: <ul style="list-style-type: none"><li><b>all</b>: Removes or redacts the specified field from the Request body, the Response body, and the activity logs.</li><li><b>request</b>: Removes or redacts the specified field from the Request body.</li><li><b>response</b>: Removes or redacts the specified field from the Response body.</li><li><b>logs</b>: Removes or redacts the specified field from the activity logs.</li></ul> The default value is <b>all</b> .  Optionally click Add item to specify additional values.	string

Tip: You can optionally click Add item to specify XPath expressions for additional fields that you want to remove or redact from the Request body, Response body, and logs.

- [Constructing XPath expressions to redact fields](#)

To define a field for redaction when using the DataPower Gateway (v5 compatible), you supply an XPath expression that specifies the field that you want to redact. If your API requests and responses use XML format, you can base your XPath statements directly on the XML content. If your API requests and responses use JSON format, you must use the DataPower XML representation of the JSON content, JSONx, to construct your XPath expression.

## Related tasks

- [Creating a new REST OpenAPI definition](#)

## Related reference

- [API Connect context variables](#)

DataPower Gateway (v5 compatible) only

## Constructing XPath expressions to redact fields

To define a field for redaction when using the DataPower® Gateway (v5 compatible), you supply an XPath expression that specifies the field that you want to redact. If your API requests and responses use XML format, you can base your XPath statements directly on the XML content. If your API requests and responses use JSON format, you must use the DataPower XML representation of the JSON content, JSONx, to construct your XPath expression.

## About this task

The following sections provide examples for constructing XPath expressions to redact a field; examples are provided for API responses in both XML and JSON format:

## XPaths for XML

Consider the following example of an XML response:

```
<xml>
 <primaryAddress>
 <streetAddress>21 2nd Street</streetAddress>
 <city>New York</city>
 <state>NY</state>
 <postalCode>10021</postalCode>
 </primaryAddress>
 <secondaryAddress>
 <streetAddress>412 Brooklyn Avenue</streetAddress>
 <city>New Jersey</city>
 <state>NJ</state>
 <postalCode>12302</postalCode>
 </secondaryAddress>
</xml>
```

To redact "streetAddress" in the preceding example the XPath expression would be: `//streetAddress` where the components of the expression have the following meaning:

Expression	Meaning
<code>//</code>	search anywhere in the XML structure
<code>//streetAddress</code>	search anywhere in the XML structure for an element of type "streetAddress"

The XML response that results from applying this XPath expression is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xml>
 <primaryAddress>
 <streetAddress>*****</streetAddress>
 <city>New York</city>
 <state>NY</state>
 <postalCode>10021</postalCode>
 </primaryAddress>
 <secondaryAddress>
 <streetAddress>*****</streetAddress>
 <city>New Jersey</city>
 <state>NJ</state>
 <postalCode>12302</postalCode>
 </secondaryAddress>
</xml>
```

Note: Both incidences of "streetAddress" have been redacted.

To redact one specific incidence of "streetAddress" in the initial example, the XPath expression would be: `//secondaryAddress/streetAddress` where the components of the expression have the following meaning:

Expression	Meaning
<code>//secondaryAddress</code>	search anywhere in the XML structure for an element of type "secondaryAddress"
<code>//secondaryAddress/streetAddress</code>	search under the preceding element for a child element of type "streetAddress"

The XML response that results from applying this XPath expression is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xml>
 <primaryAddress>
 <streetAddress>21 2nd Street</streetAddress>
 <city>New York</city>
 <state>NY</state>
 <postalCode>10021</postalCode>
 </primaryAddress>
 <secondaryAddress>
 <streetAddress>*****</streetAddress>
 <city>New Jersey</city>
 <state>NJ</state>
 <postalCode>12302</postalCode>
 </secondaryAddress>
</xml>
```

Note: Only the secondary address has been redacted.

XPaths for JSON

Consider the following example of a JSON response:

```
{
 "primaryAddress": {
 "streetAddress": "21 2nd Street",
 "city": "New York",
 "state": "NY",
 "postalCode": 10021
 },
 "secondaryAddress": {
 "streetAddress": "412 Brooklyn Avenue",
 "city": "New Jersey",
 "state": "NJ",
 "postalCode": 12302
 }
}
```

Unlike the preceding XML example, to construct an XPath for this example you must use the corresponding DataPower XML representation, JSONx. The JSONx equivalent of this JSON response is as follows:

```
<json:object xsi:schemaLocation="http://www.datapower.com/schemas/json jsonx.xsd"
xmlns:json="http://www.ibm.com/xmlns/prod/2009/jsonx" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <json:object name="primaryAddress">
 <json:string name="streetAddress">21 2nd Street</json:string>
 <json:string name="city">New York</json:string>
 <json:string name="state">NY</json:string>
 <json:number name="postalCode">10021</json:number>
 </json:object>
 <json:object name="secondaryAddress">
 <json:string name="streetAddress">412 Brooklyn Avenue</json:string>
 <json:string name="city">New York</json:string>
 <json:string name="state">NY</json:string>
 <json:number name="postalCode">10021</json:number>
 </json:object>
</json:object>
```

To redact the element "streetAddress" from the preceding JSONx structure, the XPath expression would be: `//*[ @name='streetAddress' ]` where the components of the expression have the following meaning:

Expression	Meaning
<code>//*[</code>	find any element anywhere in the structure
<code>[@name='streetAddress' ]</code>	this element has a 'name' property with value "streetAddress"

The JSON response that results from applying this XPath expression is as follows:

```
{
 "primaryAddress": {
 "streetAddress": "*****",
 "city": "New York",
 "state": "NY",
 "postalCode": 10021
 },
 "secondaryAddress": {
 "streetAddress": "*****",
 "city": "New York",
 "state": "NY",
 "postalCode": 10021
 }
}
```

Note: All incidences of "streetAddress" have been redacted in the preceding JSON structure.

To redact a specific occurrence of the "streetAddress" element, the XPath expression would be: `//*[ @name='secondaryAddress' ]/*[@name='streetAddress' ]` where the components of the expression have the following meaning:

Expression	Meaning
<code>//*[</code>	find any element anywhere in the structure
<code>//*[ @name='secondaryAddress' ]</code>	find an element anywhere in the structure that is named "secondaryAddress"
<code>//*[ @name='secondaryAddress' ]/*[@name='streetAddress' ]</code>	find a child element of any type (/* ) where the child element has a name of "streetAddress"

The JSON response that results from applying this XPath expression is as follows:

```
{
 "primaryAddress": {
 "streetAddress": "21 2nd Street",
 "city": "New York",
 "state": "NY",
 "postalCode": 10021
 },
 "secondaryAddress": {
 "streetAddress": "*****",
 "city": "New York",
 "state": "NY",
 "postalCode": 10021
 }
}
```

Note: Only the street address for the secondary address has been redacted.

## What to do next

See [Including elements in your OpenAPI 2.0 API assembly](#), [Including elements in your OpenAPI 2.0 API assembly](#) to learn how to apply a redact policy using XPath.

## Set Variable

Use the Set Variable policy to set the value of a runtime variable, or to clear a runtime variable, or to add a header variable.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [set-variable](#).

- [Configuring the Set Variable policy for DataPower API Gateway](#)  
Follow these steps to configure the Set Variable policy for DataPower API Gateway in the assembly user interface.
- [Configuring the Set Variable policy for DataPower Gateway \(v5 compatible\)](#)  
Follow these steps to configure the Set Variable policy for DataPower Gateway (v5 compatible) in the assembly user interface.

## Related tasks

- [Creating a new REST OpenAPI definition](#)

## Related reference

- [API Connect context variables](#)



# Configuring the Set Variable policy for DataPower API Gateway

Follow these steps to configure the Set Variable policy for DataPower® API Gateway in the assembly user interface.

## About this task

Note: This topic describes the Set Variable policy implementation in the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), see [Configuring the Set Variable policy for DataPower Gateway \(v5 compatible\)](#). For more information about the different types of gateway, see [API Connect gateway types](#). For information about how to configure the policy in your OpenAPI source, see [set-variable](#).

## Procedure

1. In the navigation pane, click Develop, then select the APIs tab.  
The Develop page opens.
2. Click the title of the API that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.  
For more information about working with the assembly editor for an API, see [The assembly editor](#).
4. Find the Set Variable policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. Set Variable policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is <b>set-variable</b> .	string
Description	No	A description of the policy.	string
Action	Yes	Defines what action to apply on a runtime variable. Choose one of the following values: <ul style="list-style-type: none"> <li>• Set: Indicates that you want to set a runtime variable to a string value. Can be used to set new headers or to override existing values.</li> <li>• Add: Indicates that you want to add a header variable. Can be used only to set new headers or to append a new entry of the same header name.</li> <li>• Clear: Indicates that you want to delete a runtime variable. Can be used to remove a header when the data is processed in the assembly flow.</li> </ul> The default value is Set.	string
Set, Add, or Clear	Yes	Specifies the name of the variable that you want to set, add or clear, depending on the selected Action.	string
Type	Yes	Select the data type of the variable. Choose one of the following values: <ul style="list-style-type: none"> <li>• any</li> <li>• string</li> <li>• number</li> <li>• boolean</li> </ul> For all values other than any, the value is validated against the specified data type.	string
Value	Yes*	Allocates this value to the specified variable. Can be a literal value, or another variable. * Value is required only when <b>Set</b> or <b>Add</b> is specified as the action.  For example, to <b>set</b> a named variable of <b>billing-hostname</b> to a literal value, you can specify the Value as <b>acme.com</b> .  As another example, to <b>set</b> a named variable to the value of the Content-Type header in a request, you can specify the Value entry as <b>\$(request.headers.content-type)</b> .  If the selected value of the Type field is boolean, select the Value check box to indicated a value of <b>true</b> .  Note: You can only set single string elements. Values are retrieved as strings and therefore you cannot clone a complete nodeset.	string

6. Specify a version for the policy by clicking the Source icon , and completing the **version** section of the policy YAML. For example:

```
execute:
- set-variable:
```

```

version: 2.0.0
title: set-variable
...

```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

## Example

For examples, see [set-variable](#).

DataPower Gateway (v5 compatible) only

## Configuring the Set Variable policy for DataPower Gateway (v5 compatible)

Follow these steps to configure the Set Variable policy for DataPower® Gateway (v5 compatible) in the assembly user interface.

### About this task

Note: This topic describes the Set Variable policy implementation in the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway, see [Configuring the Set Variable policy for DataPower API Gateway](#). For more information about the different types of gateway, see [API Connect gateway types](#). For details on how to configure the policy in your OpenAPI source, see [set-variable](#).

### Procedure



1. In the navigation pane, click  Develop, then select the APIs tab.  
The Develop page opens.
2. Click the title of the API that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.  
For more information about working with the assembly editor for an API, see [The assembly editor](#).
4. Find the Set Variable policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. Set Variable policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is <b>set-variable</b> .	string
Description	No	A description of the policy.	string
Action	Yes	Defines what action to apply on a runtime variable. Choose one of the following values: <ul style="list-style-type: none"> <li>• Set: Indicates that you want to set a runtime variable to a string value. Can be used to set new headers or to override existing values.</li> <li>• Add: Indicates that you want to add a header variable. Can be used only to set new headers or to append a new entry of the same header name.</li> <li>• Clear: Indicates that you want to delete a runtime variable. Can be used to remove a header when the data is processed in the assembly flow.</li> </ul> The default value is Set.	string
Set, Add, or Clear	Yes	Specifies the name of the variable that you want to set, add or clear, depending on the selected Action.	string
Type	Yes	Select the data type of the variable. Choose one of the following values: <ul style="list-style-type: none"> <li>• string</li> <li>• number</li> <li>• boolean</li> </ul> The value is validated against the specified data type.	string
Value	Yes*	Allocates this value to the specified variable. Can be a literal value, or another variable. * Value is required only when <b>Set</b> or <b>Add</b> is specified as the action.  For example, to <i>set</i> a named variable of <b>billing-hostname</b> to a literal value, you can specify the Value as <b>acme.com</b> .  As another example, to <i>set</i> a named variable to the value of the Content-Type header in a request, you can specify the Value entry as <b>\$(request.headers.content-type)</b> .  If the selected value of the Type field is boolean, select the Value check box to indicated a value of <b>true</b> .  Note: You can only set single string elements. Values are retrieved as strings and therefore you cannot clone a complete nodeset.	string

6. Specify a version for the policy by clicking the Source icon , and completing the **version** section of the policy YAML. For example:

```

execute:
- set-variable:
 version: 1.0.0
 title: set-variable
...

```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

## Example

For examples, see [set-variable](#).

## User Security

Use the User Security policy to extract a user's credentials, authenticate those credentials, and obtain authorization from the user.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [user-security](#).

## About

When you define an assembly user security action, you can define the processing for identity-extraction, authentication, and authorization or you can selectively disable any of these this aspects of processing. When disabled, this processing aspect is skipped.

When identity-extraction is enabled, the following methods are supported.

- Use basic authentication, which requires no additional configuration.
- Use context variables. For this method, specify which variable contains the user name and password.
- Use a redirect. For this method, specify the URL fragment to redirect to, and the time allowed to process.
- Use an HTML login form. For this method, specify whether to use the default or custom form and the time allowed to submit the form. For a custom form, specify the location of the form and the TLS client profile to secure the connection to the remote server.

When authentication is enabled, the following methods are supported.

- Contact an LDAP server. For this method, specify which server to contact.
- Send a request to an authentication endpoint. For this method, specify the URL of the endpoint, the TLS client profile to secure the connection, the pattern to select which response header to add, and the response header that contains the authenticated credentials.

When authorization is enabled, the following methods are supported.

- Implicitly accept any previously authenticated users, which requires no additional configuration.
- Use an HTML authorization form. For this method, specify whether to use the default or custom form and the time allowed to submit the form. For a custom form, specify the location of the form and the TLS client profile to secure the connection to the remote server.

You can attach this policy to the REST API flow.

## Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. User Security policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is <code>user-security</code> .	string
Description	No	A description of the policy.	string
Factor ID	No	The identity that identifies the results of factor-authentication in the API context.	string



Property label	Required	Description	Data type
Extract Identity Settings	Yes	<p>Select the method that is used to extract the user credentials. The following options are available:</p> <p><b>Basic</b> Use basic authentication; no additional configuration is required.</p> <p><b>Context Variable</b> The credentials are provided by API Connect context variables; specify the following properties:</p> <ul style="list-style-type: none"> <li>• Username content variable: the context variable that is used to obtain the user name.</li> <li>• password context variable: the context variable that is used to obtain the password.</li> </ul> <p><b>HTML Form</b> Use forms based identity-extraction. Select whether to use the default form or a custom form. For a custom form, specify the following properties:</p> <ul style="list-style-type: none"> <li>• Custom form endpoint: the location of the form.</li> <li>• Custom form TLS profile: the TLS client profile that is used to secure the connection to the remote server.</li> </ul> <p>In the HTML form time limit field, specify the time allowed to submit the form.</p> <p><b>Redirect</b> Use a redirect for identity-extraction; specify the following properties:</p> <ul style="list-style-type: none"> <li>• Redirect URL: the URL fragment to which to redirect the request to obtain user credentials.</li> <li>• Redirect time limit: the time allowed for the transaction to complete.</li> </ul> <p><b>Disabled</b> Identity-extraction is disabled; this aspect of processing is skipped.</p> <p>Select <b>Stop on error</b> to halt assembly processing in the event of identity-extraction failure.</p>	string
Authenticate User Settings	Yes	<p>Select the authentication method. The following options are available:</p> <p><b>Authentication URL</b> The credentials are authenticated by an external endpoint; specify the following properties:</p> <ul style="list-style-type: none"> <li>• Authentication URL: the URL of the authentication endpoint.</li> <li>• Authentication TLS profile: the TLS client profile that is used to secure the connection to the authentication endpoint.</li> <li>• Authentication response header pattern: the pattern that is used to select which response headers to add to the API context.</li> <li>• Authentication response header credential: the response header that contains the authenticated user credentials.</li> </ul> <p><b>LDAP</b> The credentials are authenticated by an LDAP user registry; from the LDAP registry list, select the required registry.</p> <p><b>Disabled</b> Authentication is disabled; this aspect of processing is skipped.</p> <p>Select <b>Stop on error</b> to halt assembly processing in the event of authentication failure.</p>	string
Authorize User Settings	Yes	<p>Select the authorization method. The following options are available:</p> <p><b>authenticated</b> Implicitly accept any previously authenticated users; no additional configuration is required.</p> <p><b>HTML Form</b> The user provides authorization through an HTML form. Select whether to use the default form or a custom form. For a custom form, specify the following properties:</p> <ul style="list-style-type: none"> <li>• Custom form endpoint: the location of the form.</li> <li>• Custom form TLS profile: the TLS client profile that is used to secure the connection to the remote server.</li> </ul> <p>In the Dynamic table entries field, enter the name of a context variable that specifies the scopes that are to be added automatically to the authorization consent form.</p> <p>In the HTML form time limit field, specify the time allowed to submit the form.</p> <p><b>Disabled</b> Authorization is disabled; this aspect of processing is skipped.</p> <p>Select <b>Stop on error</b> to halt assembly processing in the event of authorization failure.</p>	string

## Related tasks

- [Creating a new REST OpenAPI definition](#)

## Related reference

- [API Connect context variables](#)



## Validate - DataPower API Gateway

Use the Validate policy to validate the payload in an assembly flow against a schema.

### Gateway support

Note: This page describes the Validate policy implementation in the DataPower® API Gateway. If you are using the DataPower Gateway (v5 compatible), see [Validate - DataPower Gateway \(v5 compatible\)](#).

For information on the different types of gateway, see [API Connect gateway types](#).

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower API Gateway	2.0.0
	2.1.0 (DataPower API Gateway Version 10.0.1.0 or later)
	2.2.0 (DataPower API Gateway Version 10.0.2.0 or later)
	2.3.0 (DataPower API Gateway Version 10.0.2.0 or later)
	2.4.0 (DataPower API Gateway Version 10.0.4.0 or later)
	2.5.0 (DataPower API Gateway Version 10.5.0.0 or later)
	2.6.0 (DataPower API Gateway Version 10.5.0.2 or later)
	2.7.0 (DataPower API Gateway Version 10.5.0.3 or later)

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [validate - DataPower API Gateway](#).

### About

Position this policy where required in the assembly flow as follows:

- To validate the original input, position a Validate policy at the start of your flow.
- To validate an intermediate response that is returned from other invoke actions or tasks, position a Validate policy after those actions or tasks.
- To validate the response that is returned to the client application, position a Validate policy after the task that collates the response.

Note: With the DataPower API Gateway, the input to the Validate policy must be parsed data. One way to produce parsed data is to use a [Parse](#) policy before a Validate policy in your assembly flow, which provides explicit control of the parse action.

### Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Validate policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is <b>validate</b> .	string
Description	No	A description of the policy.	string
Input	No	Specifies the name of a variable in the API context. The content of the <b>body</b> field of the variable, which is represented by <b>variable_name.body</b> , is the input data to validate. By default, the variable name is <b>message</b> .	string
Output	No	Specifies the name of a variable in the API context. <ul style="list-style-type: none"> <li>• If the validation passes, the body field of the output variable, which is represented by <b>variable_name.body</b>, stores the output of the assembly validate action.</li> <li>• If the schema to validate is a JSON schema, the validation also adds any default values that are missing from the payload.</li> <li>• If the validation fails, no output is stored.</li> <li>• If an output variable is not specified, the results of the assembly validate action are not saved.</li> </ul>	string

Property label	Required	Description	Data type
Validate against	Yes	<p>Specifies the schema to be used for validating the payload. Select one of the following values:</p> <ul style="list-style-type: none"> <li><b>definition:</b> Select this value if you want to use a previously defined schema to validate the payload that is returned from other invoke actions or tasks in the assembly flow. From the Definition Object list, select the required schema. For information on defining a schema for an OpenAPI 2.0 API definition, see <a href="#">Defining schemas for an API</a>. For information on defining a schema for an OpenAPI 3.0 API definition, see <a href="#">Defining schema components</a>.</li> <li><b>URL:</b> the schema is identified by a URL location. <ul style="list-style-type: none"> <li>In the JSON Schema URL field, enter the URL of the JSON schema to be used for validating a JSON payload.</li> <li>From the XML Validation Mode list, select how XML validation is to be performed: <ul style="list-style-type: none"> <li><b>None:</b> no XML payload validation is required.</li> <li><b>xsd:</b> an XML schema will be used to validate an XML payload. In the XML Schema URL field, enter the URL of the XML schema.</li> <li><b>wsdl:</b> a WSDL schema will be used to validate a SOAP payload. In the WSDL URL field, enter the URL of the WSDL schema.</li> <li><b>soap-body:</b> validate the body of a SOAP message against the XML schema only.</li> </ul> </li> </ul> <p>Note: The following limitations apply to schemas used for JSON validation, which include JSON schemas and OpenAPI documents that are used as schemas for validation. Exceeding these limits can impact compilation performance and is not supported.</p> <ul style="list-style-type: none"> <li>Maximum of 6,500 lines. Each key and each item in an array count as a line.</li> <li>Maximum recursion depth of 250.</li> <li>Maximum of 3,000 items in <b>enum</b> lists.</li> </ul> </li> <li><b>WSDL:</b> (available with a SOAP service based API only) use the XML schema in the WSDL file associated with the API operation or the API path.</li> <li><b>body-param:</b> validate the request input against the schema definition that is specified in the TYPE field for the request parameter for this operation. For information about how to create a request parameter, see <a href="#">Defining parameters for an operation</a>.</li> <li><b>response-param:</b> validate the response to be returned to the client application, against the schema definition that is specified in the SCHEMA field for the response parameter for this operation. For information about how to create a response parameter, see <a href="#">Defining responses for an operation</a>.</li> <li><b>GraphQL:</b> (available for a GraphQL proxy API only) validate the payload against the GraphQL schema that has been imported into the GraphQL proxy API. In addition, either the GraphQL query or response, depending on the input, is analyzed against the GraphQL schema to calculate the cost, and the result is placed in the API context. For more information on GraphQL proxy APIs, see <a href="#">Creating a GraphQL proxy API</a> and <a href="#">GraphQL context variables</a>.</li> </ul>	string
XSLT version	No	The XSLT processor version. The default value is XSLT10.	string
Strict	No	Whether to enable strict XSLT error checking. Non-strict operations attempt to recover from certain errors, such as use of undeclared variables, calling undeclared templates, and so forth. By default, strict XSLT error checking is enabled.	boolean
Profile rule	No	Whether to enable stylesheet profiling. This option should not be used in production environments. By default, stylesheet profiling is disabled.	boolean
Debug rule	No	Whether to run the stylesheet, XQuery script, and JSONiq script in debug mode. When a stylesheet, XQuery script, or JSONiq script is run in debug mode, it generates a custom web page instead of displaying its normal output. The web page details exactly what occurred during execution, including the values of variables and where particular pieces of the output came from. This option should not be used in production environments. By default, debug mode is disabled.	boolean
Streaming rule	No	Whether the stylesheet must be run in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, an error is generated and the input is not processed. By default, streaming mode is disabled.	boolean
Attempt streaming rule	No	Whether to attempt to run the stylesheet in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, a warning is generated during compilation and the stylesheet is read in the entire input as normal at execution time. By default, attempting to run the stylesheet in streaming mode is disabled.	boolean
Minimum output escaping rule	No	Whether to escape output produced from the stylesheet during processing. Minimal escaping is particularly useful when handling non-English character sets. By default, minimum escaping is disabled.	boolean
Maximum stack size	No	The maximum number of bytes that the stack is allowed to use while executing a stylesheet or other compiled content. This setting is used to block infinite recursion. The minimum value is 10 kilobytes, or 10,240 bytes. The maximum value is 100 megabytes, or 104,857,600 bytes. The default value is 1 megabyte, or 1,048,576 bytes.	integer
WS-I Basic Profile validation	No	<p>The validation behavior to apply to WSDL files that are checked for conformance to section 5 of WS-I Basic Profile (version 1.0, April 2004). The default setting is Warn.</p> <p>Ignore Disables conformance checking.</p> <p>Warn Logs warnings for violations.</p> <p>Fail Forces conformance. Fails if the file contains any violation.</p>	string

Property label	Required	Description	Data type
Validate message body	No	The validation behavior for the <code>soap:Body</code> . The default setting is Strict.  Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition.  Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match.  Skip Disables validation of this part of the message.	string
Validate message headers	No	The validation behavior for the <code>soap:Header</code> . The default setting is Lax.  Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition.  Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match.  Skip Disables validation of this part of the message.	string
Validate message fault details	No	Specifies the validation behavior for the fault detail. The default setting is Strict.  Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition.  Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match.  Skip Disables validation of this part of the message.	string
Require wrappers on fault details specified by type	No	Whether to require compatibility with RPC-style wrappers. By default, RPC-style wrappers are not required.	boolean
Specifically allow <code>xsi:type='SOAP-ENC:Array'</code> rule	No	Whether to allow the schema to accept most uses of elements with <code>xsi:type='SOAP-ENC:Array'</code> consistent with SOAP 1.1 Section 5, even when these attributes violate the XML Schema specification. Normally, the <code>xsi:type</code> attribute must name a type equal to or derived from the actual type of the element. For schemas compiled with this option, <code>xsi:type</code> is accepted specifically for the SOAP 1.1 Encoding 'Array' complex type if the element type is derived from <code>SOAP-ENC:Array</code> . The opposite is the normal allowable case. By default, elements with <code>xsi:type='SOAP-ENC:Array'</code> are not accepted.	boolean
Validate SOAP 1.1 encoding rule	No	Whether to perform extra schema validation following the encoding rules in SOAP 1.1 Section 5. When enabled, members of SOAP arrays are validated, attributes such as <code>@id</code> and <code>@href</code> are allowed even if they are not allowed by the schema, and <code>@href</code> values are checked to ensure that they have a corresponding <code>@id</code> element. By default, the extra validation is not performed.	boolean
Wildcards ignore <code>xsi:type</code> rule	No	Whether <code>xs:any</code> elements in the schema validate only child elements by name. The XML Schema specification requires that, if a wildcard matches an element but that element does not have an element declaration, the element is instead validated according to an <code>xsi:type</code> attribute on it. This option ignores those <code>xsi:type</code> attributes. It should be used for cases such as SOAP envelope validation where a further validation step will validate the contents matching the wildcard, possibly using the SOAP 1.1 encoding rules. By default, <code>xsi:type</code> attributes are not ignored.	boolean
Strict SOAP envelope version	No	Whether to strictly follow the SOAP binding in the WSDL. When enabled, only messages bound to SOAP 1.2 appear in SOAP 1.2 envelopes and only messages bound to SOAP 1.1 appear in SOAP 1.1 envelopes. By default, strict SOAP binding is disabled.	boolean
Debug XACML policy	No	Whether to compile XACML policies with debug information. Note that the XACML debugging messages are also controlled by the log event in the XACML category. Use the debug log level to view the full XACML debugging messages. By default, XACML policies are not compiled with debug information.	boolean
Accept MTOM/XOP optimized messages	No	Specifies whether the schema or WSDL document accepts messages where base64-encoded binary content was optimized according to the MTOM/XOP specifications. XOP binary-optimization replaces base64-encoded binary data with an <code>xop:Include</code> reference element that references the unencoded binary data located in an attachment. By default, MTOM/XOP optimized messages are disabled.  <ul style="list-style-type: none"> <li>When disabled, such optimized messages are rejected by validation of the optimized form. Rejection occurs because the schema specifies a simple type that accepts base64-encoded data, such as <code>xs:base64Binary</code> or <code>xs:string</code>, but the message contains an <code>xop:Include</code> element instead.</li> <li>When enabled, an <code>xop:Include</code> element can optionally appear in place of content for any XML Schema simple type that validates base64-encoded binary data. The <code>xop:Include</code> element itself will be validated according to the built-in schema in <code>store:///schemas/xop.xsd</code>.</li> </ul>	boolean

For examples, see [validate - DataPower API Gateway](#).

## Related tasks

- [Defining Paths for a REST API](#)
- [Creating a new REST OpenAPI definition](#)

DataPower Gateway (v5 compatible)

## Validate - DataPower Gateway (v5 compatible)

Use the Validate policy to validate the payload in an assembly flow against a JSON or an XML schema.

### Gateway support

Note: This topic describes the Validate policy implementation in the DataPower® Gateway (v5 compatible). If you are using the DataPower API Gateway, see [Validate - DataPower API Gateway](#).

For information on the different types of gateway, see [API Connect gateway types](#).

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower Gateway (v5 compatible)	1.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [validate - DataPower Gateway \(v5 compatible\)](#).

Restriction:

- The schema that represents the XML can reference only one XML namespace.
- The schema cannot reference polymorphic XML elements.
- The validation works on the `message.body` variable and not any other output/context variable. If the invoke policy contains a configured response object variable, then `message.body` is not set, and validate is not able to act.
- If you use the `multipleOf` keyword in a schema definition for the API then, due to rounding behavior, the specified value must satisfy the following conditions, otherwise the validation fails when the API is called:
  - The value must not be less than 0.0000009999999999999999848869.
  - If the value is greater than 1, the amount before the decimal point must not be greater than 999999999999999934463.

### About

You can attach this policy to the following API flow:

- REST

Position this policy where required in the assembly flow as follows:

- To validate the original input, position a Validate policy at the start of your flow.
- To validate an intermediate response that is returned from other invoke actions or tasks, position a Validate policy after those actions or tasks.
- To validate the response that is returned to the client application, position a Validate policy after the task that collates the response.

You can apply a different OpenAPI schema definition to each Validate policy either by choosing from the set of schema definitions that is specified at the API level, or the schema definition at the operation level.

### Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Validate policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is <code>validate</code> .	string
Description	No	A description of the policy.	string
Definition	Yes	Specifies the schema definition to be used for validating the payload. Valid values: <ul style="list-style-type: none"> <li>• <code>request</code>: Select this value to validate the request input against the schema definition that is specified in the TYPE field for the request parameter for this operation. For information about how to create a request parameter, see <a href="#">Configuring an operation</a>.</li> <li>• <code>response</code>: Select this value to validate the response to be returned to the client application, against the schema definition that is specified in the SCHEMA field for the response parameter for this operation. For information about how to create a response parameter, see <a href="#">Configuring an operation</a>.</li> <li>• <code>#/definitions/definition_name</code>: Select this value for a previously defined schema definition to be used to validate the payload that is returned from other invoke actions or tasks in the assembly flow. For information on defining a schema for an API definition, see <a href="#">Editing an API definition</a>.</li> </ul>	string

### Related tasks

- [Defining Paths for a REST API](#)
- [Creating a new REST OpenAPI definition](#)

## Validate Username Token

Use the Validate Username Token policy to validate a Web Services Security (WS-Security) UsernameToken in a SOAP payload before allowing access to the protected resource.

### Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0 1.1.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [validate-username-token](#).

### About

A WS-Security UsernameToken enables a user identity to be passed securely over a multi-point message path. The Validate Username Token policy extracts the UsernameToken element from the request payload, authenticates the extracted username and password, and provides access to the protected resource based on the authentication result. The policy has two authentication methods: Lightweight Directory Access Protocol (LDAP) user registry, or Authentication URL.

The Validate Username Token policy supports both passwordText and passwordDigest types of password. When the authentication method is Authentication URL with a passwordDigest, a basic authentication header that contains a Base64 encoded username and passwordDigest is sent to the URL. In addition, a custom header named X-IBM-PasswordType is set with a value of digest. The following table shows the authentication process based on password type:

Table 2. Authentication URL process by password type

passwordText	passwordDigest
Authentication: Basic base64(username:password)	Authentication: Basic base64(username:passwordDigest) X-IBM-PasswordType: 'digest'

Position this policy where required in the assembly flow as follows:

- To validate the original input, position a Validate Username Token policy at the start of your flow.
- To validate an intermediate response that is returned from other invoke actions or tasks, position a Validate Username Token policy after those actions or tasks.
- To validate the response that is returned to the client application, position a Validate Username Token policy after the task that collates the response.

### Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 3. Validate Username Token policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is <b>validate-username-token</b> .	string
Description	No	A description of the policy.	string
Authentication type (policy version 1.0.0 only)	Yes	The authentication type to use to validate the UsernameToken. Valid values: <ul style="list-style-type: none"> <li>• <b>Authentication URL</b>: Select this value to validate the user credentials against an authentication URL.</li> <li>• <b>LDAP registry</b>: Select this value to validate the user credentials against an LDAP user registry.</li> </ul> The default value is: <b>Authentication URL</b> .	string
Authentication URL (policy version 1.0.0 only)	Yes	The authentication URL to use to validate the UsernameToken user credentials against. Note: This property is required only if Authentication type is set to <b>Authentication URL</b> .	string
TLS profile (policy version 1.0.0 only)	No	The TLS profile to use for the secure transmission of data to the authentication URL. Note: This property is available only if Authentication type is set to <b>Authentication URL</b> .	string
LDAP registry name (policy version 1.0.0 only)	Yes	The name of the LDAP user registry to validate the UsernameToken user credentials against. You can select a name from the drop-down list, or type a name manually. Note: This property is required only if Authentication type is set to <b>LDAP registry</b> .	string
User Registry Name (policy version 1.1.0 and later)	Yes	Select the LDAP or Authentication URL registry to use to validate the UsernameToken.	string
LDAP search attribute <sup>1</sup>	Yes	The name of the LDAP user password attribute. Note: This property is required only for an LDAP user registry.	string

### Examples

The following example shows an LDAP user registry authentication:

```
- validate-usnametoken:
 version: 1.0.0
 title: "validate-usnametoken"
 auth-type: "LDAP Registry"
 ldap-registry: "wstest"
 ldap-search-attribute: "userPassword"
```

The following example shows an Authentication URL definition:

```
- validate-usnametoken:
 version: 1.0.0
 title: "validate-usnametoken"
 auth-type: "Authentication URL"
 auth-url: "https://www.google.com"
 tls-profile: "default-ssl-profile"
```

## Errors

The policy returns an HTTP 200 status code when successful, and the input payload is copied to the output flow. For all failure types the policy returns an HTTP 500 status code, and the output contains the SOAP fault.

Tip: If there are authentication failures, try verifying the LDAP user registry configuration as follows:

- Ensure Search (DN) is set as the communication method.
- Ensure Authenticated Bind is set so that specific permissions are required to search the registry.
- Ensure the Admin DN and Password fields are correctly completed for the Distinguished Name (DN) of a user authorized to carry out searches in the LDAP directory.
- Ensure that a combination of Base DN, Prefix, and Suffix are set, such that they fully describe the user DN. For example:
  - For a user named: `cn=alice, dc=ibm, dc=com`

```
BaseDN: dc=ibm
Prefix: cn=alice
Suffix: dc=com
```

where the user DN is calculated as: Prefix + BaseDN + Suffix.

## Related concepts

- [LDAP authentication](#)

## Related tasks

- [Creating a new REST OpenAPI definition](#)

## Related information

- [Creating an LDAP registry](#)
- [Authenticating by using your enterprise user registry](#)

<sup>1</sup> When authenticating with LDAP and passwordText, the policy uses the username and password as LDAP bind credentials. However, when authenticating with LDAP and passwordDigest, the digest itself cannot be used for authentication. Instead, an LDAP search for the username is performed by using the administrator's distinguished name (DN) and password, and an attribute corresponding to the contents of the ldap-search-attribute is retrieved. A hash of the contents of this attribute (along with the Nonce and Created attributes, as in the WS-Security UsernameToken profile specification) is then compared to the passwordDigest.

## Websocket Upgrade

Use the Websocket Upgrade policy to process API requests and responses through a WebSocket connection.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

- [Configuring the Websocket Upgrade policy for DataPower API Gateway](#)  
Follow these steps to configure the Websocket Upgrade policy for DataPower API Gateway in the assembly user interface.
- [Configuring the Websocket Upgrade policy for DataPower Gateway \(v5 compatible\)](#)  
Follow these steps to configure the Websocket Upgrade policy for DataPower Gateway (v5 compatible) in the assembly user interface.

API Gateway only

## Configuring the Websocket Upgrade policy for DataPower API Gateway

Follow these steps to configure the Websocket Upgrade policy for DataPower® API Gateway in the assembly user interface.

## About this task

Note: This topic describes the Websocket Upgrade policy implementation in the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), see [Configuring the Websocket Upgrade policy for DataPower Gateway \(v5 compatible\)](#). For more information about the different types of gateway, see [API Connect gateway types](#).

For information about how to configure the policy in your OpenAPI source, see [websocket-upgrade](#).

## Procedure



1. In the navigation pane, click  Develop, then select the APIs tab.  
The Develop page opens.
2. Click the title of the API that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.  
For more information about working with the assembly editor for an API, see [The assembly editor](#).
4. Find the Websocket Upgrade policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. Websocket Upgrade policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is <b>websocket-upgrade</b> .	string
Description	No	A description of the policy.	string
URL	Yes	Specify the URL to be invoked.	string
TLS Profile	No	Specifies a TLS profile to use for the secure transmission of data.	string
Timeout	No	The time to wait before a reply back from the endpoint (in seconds). The default value is <b>60</b> .	integer
Follow redirects	No	Specifies the behavior if the back-end server returns the HTTP status code <b>301 Moved Permanently</b> . If you select this check box, the <b>invoke</b> policy follows the URL redirection by making a further call to the URL specified in the <b>Location</b> header in the response. If you clear this check box, the <b>invoke</b> saves the <b>301</b> status code and the API call is considered to be complete. Note: The <b>follow-redirect</b> property is supported only by the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), the <b>invoke</b> always follows the URL redirection; the <b>proxy</b> policy (not supported by the DataPower API Gateway) saves the <b>301</b> status code and completes the API call without following the URL redirection.	boolean
Username	No	The username to use for HTTP Basic authentication.	string
Password	No	The password to use for HTTP Basic authentication.	string
Inject proxy headers	No	If you select this check box, the <b>invoke</b> policy injects the <b>X-Forwarded-For</b> , <b>X-Forwarded-To</b> , <b>X-Forwarded-Host</b> , and <b>X-Forwarded-Proto</b> headers to the request that is sent to the target URL. The check box is cleared by default.	boolean
Decode Request Params	No	If you select this check box, any request parameters that are referenced by a variable definition on the target URL of the invoke policy are URL-decoded. The check box is cleared by default.	boolean
QueryParam encode	No	If you select this check box, all "+" characters in the query parameter values of the target URL are encoded to %2F. The check box is cleared by default.	boolean
Header control	No	Specifies the headers in <b>message.headers</b> that you want to copy to the target URL. To prevent headers from being copied, complete the following steps: a. Select Blocklist. b. Click Add blocklist. c. In the empty field that is displayed, enter the header name. d. To add further headers, repeat the previous steps. To specify headers that you want to be copied, complete the following steps: a. Select Allowlist. b. Click Add allowlist. c. In the empty field that is displayed, enter the header name. d. To add further headers, repeat the previous steps. The values that you specify are in regular expression format. For example, to specify the Content-Type header, enter <b>^Content-Type\$</b>  By default, Blocklist is selected, with no blocklist entries, meaning that <b>all</b> headers are copied.	string



Property label	Required	Description	Data type
Parameter control	No	<p>Specifies the parameters in the incoming request that you want to be copied to the target URL. To prevent parameters from being copied, complete the following steps:</p> <ol style="list-style-type: none"> <li>Select Blocklist.</li> <li>Click Add blocklist.</li> <li>In the empty field that is displayed, enter the parameter name.</li> <li>To add further parameters, repeat the previous steps.</li> </ol> <p>To specify parameters that you want to be copied, complete the following steps:</p> <ol style="list-style-type: none"> <li>Select Allowlist.</li> <li>Click Add allowlist.</li> <li>In the empty field that is displayed, enter the parameter name.</li> <li>To add further parameters, repeat the previous steps.</li> </ol> <p>The values that you specify are in regular expression format.</p> <p>For example, if the incoming request is</p> <pre>http://apigw/org/sandbox/petstore/base?petid=100&amp;display=detailed</pre> <p>and you specify a white list entry of <code>^petid\$</code>, the target URL at run time will be</p> <pre>http://myhost/mypath?storeid=3&amp;petid=100</pre> <p>By default, Allowlist is selected, with no allowlist entries, meaning that <b>no</b> parameters are copied.</p>	string
Request assembly	No	<p>The request processing assembly. For details on configuring an assembly in the <b>execute</b> section, see <a href="#">execute</a>. For example:</p> <pre>request-assembly:   execute   .   .   .   policy assembly   .   .   .</pre>	object
Response assembly	No	<p>The response processing assembly. For details on configuring an assembly in the <b>execute</b> section, see <a href="#">execute</a>. For example:</p> <pre>response-assembly:   execute   .   .   .   policy assembly   .   .   .</pre>	object

6. Specify a version for the policy by clicking the Source icon , and completing the **version** section of the policy YAML. For example:

```
execute:
- websocket-upgrade:
 version: 2.0.0
 title: websocket-upgrade
...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

DataPower Gateway (v5 compatible) only


## Configuring the Websocket Upgrade policy for DataPower Gateway (v5 compatible)

Follow these steps to configure the Websocket Upgrade policy for DataPower® Gateway (v5 compatible) in the assembly user interface.

### About this task

Note: This topic describes the Websocket Upgrade policy implementation in the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway, see [Configuring the Websocket Upgrade policy for DataPower API Gateway](#). For more information about the different types of gateway, see [API Connect gateway types](#). For details on how to configure the policy in your OpenAPI source, see [websocket-upgrade](#).


### Procedure

- In the navigation pane, click  Develop, then select the APIs tab. The Develop page opens.
- Click the title of the API that you want to work with, or create a new API.
- Select the Gateway tab, then click Policies in the navigation pane. For more information about working with the assembly editor for an API, see [The assembly editor](#).
- Find the Websocket Upgrade policy in the palette, and drag the policy onto your canvas.

5. Specify the following properties.

Table 1. Websocket Upgrade policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is <b>websocket-upgrade</b> .	string
Description	No	A description of the policy.	string
URL	Yes	Specify the URL to be invoked.	string
TLS Profile	No	Specifies a TLS profile to use for the secure transmission of data.	string
Timeout	No	The time to wait before a reply back from the endpoint (in seconds). The default value is <b>60</b> .	integer
Follow redirects	No	Specifies the behavior if the back-end server returns the HTTP status code <b>301 Moved Permanently</b> . If you select this check box, the <b>invoke</b> policy follows the URL redirection by making a further call to the URL specified in the <b>Location</b> header in the response. If you clear this check box, the <b>invoke</b> saves the <b>301</b> status code and the API call is considered to be complete. Note: The <b>follow-redirect</b> property is supported only by the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), the <b>invoke</b> always follows the URL redirection; the <b>proxy</b> policy (not supported by the DataPower API Gateway) saves the <b>301</b> status code and completes the API call without following the URL redirection.	boolean
Username	No	The username to use for HTTP Basic authentication.	string
Password	No	The password to use for HTTP Basic authentication.	string
Inject proxy headers	No	If you select this check box, the <b>invoke</b> policy injects the <b>X-Forwarded-For</b> , <b>X-Forwarded-To</b> , <b>X-Forwarded-Host</b> , and <b>X-Forwarded-Proto</b> headers to the request that is sent to the target URL. The check box is cleared by default.	boolean
Decode Request Params	No	If you select this check box, any request parameters that are referenced by a variable definition on the target URL of the invoke policy are URL-decoded. The check box is cleared by default.	boolean
QueryParam encode	No	If you select this check box, all "+" characters in the query parameter values of the target URL are encoded to %2F. The check box is cleared by default.	boolean
Request assembly	No	The request processing assembly. For details on configuring an assembly in the <b>execute</b> section, see <a href="#">execute</a> . For example:  <pre>request-assembly:   execute   .   .   policy assembly   .   .</pre>	object
Response assembly	No	The response processing assembly. For details on configuring an assembly in the <b>execute</b> section, see <a href="#">execute</a> . For example:  <pre>response-assembly:   execute   .   .   policy assembly   .   .</pre>	object

6. Specify a version for the policy by clicking the Source icon , and completing the **version** section of the policy YAML. For example:

```
execute:
- websocket-upgrade:
 version: 1.0.0
 title: websocket-upgrade
...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

## XML to JSON

Use the XML to JSON policy to convert the context payload of your API from the extensible markup language (XML) format to JavaScript Object Notation (JSON).

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

The XML to JSON policy uses a simple convention, based on BadgerFish, to convert your API context payload from XML to JSON. The XML content is preserved, including the attributes and namespaces. No additional configuration is required. For more information about the BadgerFish convention, including some examples, see [BadgerFish](#).

Use the API Designer assembly view when you are creating your API definition to add a built-in policy to the flow.

The policy must be attached to the flow at the point at which you require the conversion to be performed. For example, if you need to convert an XML-formatted request into a JSON-formatted request, the policy must be attached to the request flow.

The policy reads input from the `message.body`, if that context exists, otherwise from the `request.body`, and then writes the output to the `message.body`.

Note: If you are using the DataPower API Gateway, the input to the XML to JSON policy must be parsed data. One way to produce parsed data is to use a [Parse](#) policy before an XML to JSON policy in your assembly flow, which provides explicit control of the parse action.

## Examples

For example, the following simple XML object

```
<a>hello
```

becomes

```
{ "a": { "$" : "hello" } }
```

The following XML object with an attribute

```
hello
```

becomes

```
{ "a": { "$" : "hello", "@type" : "world" } }
```

For examples, see [xml-to-json](#).

- [Configuring the XML to JSON policy for DataPower API Gateway](#)  
Follow these steps to configure the XML to JSON policy for DataPower API Gateway in the assembly user interface.
- [Configuring the XML to JSON policy for DataPower Gateway \(v5 compatible\)](#)  
Follow these steps to configure the XML to JSON policy for DataPower Gateway (v5 compatible) in the assembly user interface.



## Configuring the XML to JSON policy for DataPower API Gateway

Follow these steps to configure the XML to JSON policy for DataPower® API Gateway in the assembly user interface.

### About this task

Note: This topic describes the XML to JSON policy implementation in the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), see [Configuring the Set Variable policy for DataPower Gateway \(v5 compatible\)](#). For more information about the different types of gateway, see [API Connect gateway types](#). This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [xml-to-json](#).

## Procedure

1. In the navigation pane, click Develop, then select the APIs tab.  
The Develop page opens.
2. Click the title of the API that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.  
For more information about working with the assembly editor for an API, see [The assembly editor](#).
4. Find the XML to JSON policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. XML to JSON policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is <code>xml-to-json</code> .	string
Description	No	A description of the policy.	string
Input	No	The input message to convert. Specify the name of a variable in the API context. <code>variableName.body</code> , the message payload, represents the XML input to convert. The default value of the variable is <code>message</code> and <code>message.body</code> is the default input.	string
Output	No	The output message to store the conversion result. Specify the name of a variable in the API context. <code>variableName.body</code> represents the result of conversion from XML format to JSON format. When the specified input message is the default message, the default output is <code>message.body</code> . Otherwise, when the input message is the variable <code>my-message-variable</code> , for example, the default output is <code>my-message-variable.body</code> . The variable cannot be any read-only in the API context.	string
Conversion type	No	The conversion type that determines the target format of the output. The following options are available: <ul style="list-style-type: none"> <li>• badgerFish: BadgerFish convention is used to determine the target conversion format of the output.</li> <li>• apicv5: apicv5 convention is used to determine the target conversion format of the output.</li> </ul>	string

6. Specify a version for the policy by clicking the Source icon `</>`, and completing the `version` section of the policy YAML. For example:

```
execute:
- xml-to-json:
 version: 2.0.0
 title: xml-to-json
...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

## Example

For example, the following simple XML object

```
<a>hello
```

becomes

```
{ "a": { "$" : "hello" } }
```

The following XML object with an attribute

```
hello
```

becomes

```
{ "a": { "$" : "hello", "@type" : "world" } }
```

For more examples, see [xml-to-json](#).

DataPower Gateway (v5 compatible) only

## Configuring the XML to JSON policy for DataPower Gateway (v5 compatible)

Follow these steps to configure the XML to JSON policy for DataPower® Gateway (v5 compatible) in the assembly user interface.

### About this task

Note: This topic describes the XML to JSON policy implementation in the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway, see [Configuring the XML to JSON policy for DataPower API Gateway](#). For more information about the different types of gateway, see [API Connect gateway types](#). This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [xml-to-json](#).

### Procedure



1. In the navigation pane, click  Develop, then select the APIs tab.  
The Develop page opens.
2. Click the title of the API that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.  
For more information about working with the assembly editor for an API, see [The assembly editor](#).
4. Find the XML to JSON policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. XML to JSON policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is <code>xml-to-json</code> .	string
Description	No	A description of the policy.	string

6. Specify a version for the policy by clicking the Source icon , and completing the `version` section of the policy YAML. For example:

```
execute:
- xml-to-json:
 version: 1.0.0
 title: xml-to-json
...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

## Example

For examples, see [xml-to-json](#).

## XSLT

Use the XSLT policy to apply an XSLT transform to the payload of the API definition.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.4.0 or later)

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [xslt](#).

Note: If you are using the DataPower API Gateway, the input to the XSLT policy must be parsed data. One way to produce parsed data is to use a [Parse](#) policy before an XSLT policy in your assembly flow, which provides explicit control of the parse action.

For examples of the OpenAPI definitions of XSLT policies, see [XSLT policy examples](#).

For more examples of how to use XSLT to access and modify properties and context, see [Implementation code examples](#) and, if you are using the DataPower API Gateway, [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway](#).

## Errors

The following error can be thrown while the policy is being executed:

- **TransformError** - a generic error that captures all errors that occur during the execution of the policy.
- [Configuring the XSLT policy for DataPower API Gateway](#)  
Follow these steps to configure the XSLT policy for DataPower API Gateway in the assembly user interface.
- [Configuring the XSLT policy for DataPower Gateway \(v5 compatible\)](#)  
Follow these steps to configure the XSLT policy for DataPower Gateway (v5 compatible) in the assembly user interface.
- [XSLT policy examples](#)  
Examples of the OpenAPI definitions of XSLT policies.
- [Implementation code examples](#)  
Example XSLT and GatewayScript code snippets.

## Related concepts

- [Variable references in API Connect](#)

## Related tasks

- [Creating a new REST OpenAPI definition](#)

## Related reference

- [API Connect context variables](#)



# Configuring the XSLT policy for DataPower API Gateway

Follow these steps to configure the XSLT policy for DataPower® API Gateway in the assembly user interface.

## About this task

Note: This topic describes the XSLT policy implementation in the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), see [Configuring the XSLT policy for DataPower Gateway \(v5 compatible\)](#). For more information about the different types of gateway, see [API Connect gateway types](#).

For information about how to configure the policy in your OpenAPI source, see [xslt](#).

## Procedure


1. In the navigation pane, click Develop, then select the APIs tab.  
The Develop page opens.
2. Click the title of the API that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.  
For more information about working with the assembly editor for an API, see [The assembly editor](#).
4. Find the XSLT policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. XSLT policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is <b>xslt</b> .	string
Description	No	A description of the policy.	string

Property label	Required	Description	Data type
Input	No	Indicates whether this XSLT input document uses the context current payload, or if there is no input. The check box is cleared by default, which indicates that there is no input.	boolean
Serialize output	No	If you select this option, the output tree that is generated by the XSLT policy is serialized. The content of <code>message.body</code> is updated with the serialized binary data rather than the XML tree. The check box is cleared by default, which indicates that the output tree is not serialized.	boolean
Source	Yes	The XSLT transform source to execute.	string
XSLT version	No	The XSLT processor version. The default value is XSLT10.	string
Strict	No	Whether to enable strict XSLT error checking. Non-strict operations attempt to recover from certain errors, such as use of undeclared variables, calling undeclared templates, and so forth. By default, strict XSLT error checking is enabled.	boolean
Profile rule	No	Whether to enable stylesheet profiling. This option should not be used in production environments. By default, stylesheet profiling is disabled.	boolean
Debug rule	No	Whether to run the stylesheet, XQuery script, and JSONiq script in debug mode. When a stylesheet, XQuery script, or JSONiq script is run in debug mode, it generates a custom web page instead of displaying its normal output. The web page details exactly what occurred during execution, including the values of variables and where particular pieces of the output came from. This option should not be used in production environments. By default, debug mode is disabled.	boolean
Streaming rule	No	Whether the stylesheet must be run in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, an error is generated and the input is not processed. By default, streaming mode is disabled.	boolean
Attempt streaming rule	No	Whether to attempt to run the stylesheet in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, a warning is generated during compilation and the stylesheet is read in the entire input as normal at execution time. By default, attempting to run the stylesheet in streaming mode is disabled.	boolean
Minimum output escaping rule	No	Whether to escape output produced from the stylesheet during processing. Minimal escaping is particularly useful when handling non-English character sets. By default, minimum escaping is disabled.	boolean
Maximum stack size	No	The maximum number of bytes that the stack is allowed to use while executing a stylesheet or other compiled content. This setting is used to block infinite recursion. The minimum value is 10 kilobytes, or 10,240 bytes. The maximum value is 100 megabytes, or 104,857,600 bytes. The default value is 1 megabyte, or 1,048,576 bytes.	integer
WS-I Basic Profile validation	No	The validation behavior to apply to WSDL files that are checked for conformance to section 5 of WS-I Basic Profile (version 1.0, April 2004). The default setting is Warn.  Ignore Disables conformance checking.  Warn Logs warnings for violations.  Fail Forces conformance. Fails if the file contains any violation.	string
Validate message body	No	The validation behavior for the <code>soap:Body</code> . The default setting is Strict.  Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition.  Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match.  Skip Disables validation of this part of the message.	string
Validate message headers	No	The validation behavior for the <code>soap:Header</code> . The default setting is Lax.  Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition.  Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match.  Skip Disables validation of this part of the message.	string
Validate message fault details	No	Specifies the validation behavior for the fault detail. The default setting is Strict.  Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition.  Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match.  Skip Disables validation of this part of the message.	string

Property label	Required	Description	Data type
Require wrappers on fault details specified by type	No	Whether to require compatibility with RPC-style wrappers. By default, RPC-style wrappers are not required.	boolean
Specifically allow xsi:type='SOAP-ENC:Array' rule	No	Whether to allow the schema to accept most uses of elements with <code>xsi:type='SOAP-ENC:Array'</code> consistent with SOAP 1.1 Section 5, even when these attributes violate the XML Schema specification. Normally, the <code>xsi:type</code> attribute must name a type equal to or derived from the actual type of the element. For schemas compiled with this option, <code>xsi:type</code> is accepted specifically for the SOAP 1.1 Encoding 'Array' complex type if the element type is derived from <code>SOAP-ENC:Array</code> . The opposite is the normal allowable case. By default, elements with <code>xsi:type='SOAP-ENC:Array'</code> are not accepted.	boolean
Validate SOAP 1.1 encoding rule	No	Whether to perform extra schema validation following the encoding rules in SOAP 1.1 Section 5. When enabled, members of SOAP arrays are validated, attributes such as @id and @href are allowed even if they are not allowed by the schema, and @href values are checked to ensure that they have a corresponding @id element. By default, the extra validation is not performed.	boolean
Wildcards ignore xsi:type rule	No	Whether <code>xs:any</code> elements in the schema validate only child elements by name. The XML Schema specification requires that, if a wildcard matches an element but that element does not have an element declaration, the element is instead validated according to an <code>xsi:type</code> attribute on it. This option ignores those <code>xsi:type</code> attributes. It should be used for cases such as SOAP envelope validation where a further validation step will validate the contents matching the wildcard, possibly using the SOAP 1.1 encoding rules. By default, <code>xsi:type</code> attributes are not ignored.	boolean
Strict SOAP envelope version	No	Whether to strictly follow the SOAP binding in the WSDL. When enabled, only messages bound to SOAP 1.2 appear in SOAP 1.2 envelopes and only messages bound to SOAP 1.1 appear in SOAP 1.1 envelopes. By default, strict SOAP binding is disabled.	boolean
Debug XACML policy	No	Whether to compile XACML policies with debug information. Note that the XACML debugging messages are also controlled by the log event in the XACML category. Use the debug log level to view the full XACML debugging messages. By default, XACML policies are not compiled with debug information.	boolean
Accept MTOM/XOP optimized messages	No	Specifies whether the schema or WSDL document accepts messages where base64-encoded binary content was optimized according to the MTOM/XOP specifications. XOP binary-optimization replaces base64-encoded binary data with an <code>xop:Include</code> reference element that references the unencoded binary data located in an attachment. By default, MTOM/XOP optimized messages are disabled. <ul style="list-style-type: none"> <li>When disabled, such optimized messages are rejected by validation of the optimized form. Rejection occurs because the schema specifies a simple type that accepts base64-encoded data, such as <code>xs:base64Binary</code> or <code>xs:string</code>, but the message contains an <code>xop:Include</code> element instead.</li> <li>When enabled, an <code>xop:Include</code> element can optionally appear in place of content for any XML Schema simple type that validates base64-encoded binary data. The <code>xop:Include</code> element itself will be validated according to the built-in schema in <code>store:///schemas/xop.xsd</code>.</li> </ul>	boolean

6. Specify a version for the policy by clicking the Source icon , and completing the `version` section of the policy YAML. For example:

```
execute:
- xslt:
 version: 2.1.0
 title: xslt
...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

DataPower Gateway (v5 compatible) only

## Configuring the XSLT policy for DataPower Gateway (v5 compatible)

Follow these steps to configure the XSLT policy for DataPower® Gateway (v5 compatible) in the assembly user interface.

### About this task

Note: This topic describes the XSLT policy implementation in the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway, see [Configuring the XSLT policy for DataPower API Gateway](#). For more information about the different types of gateway, see [API Connect gateway types](#). For details on how to configure the policy in your OpenAPI source, see [xslt](#).

### Procedure



- In the navigation pane, click  Develop, then select the APIs tab. The Develop page opens.
- Click the title of the API that you want to work with, or create a new API.
- Select the Gateway tab, then click Policies in the navigation pane. For more information about working with the assembly editor for an API, see [The assembly editor](#).
- Find the XSLT policy in the palette, and drag the policy onto your canvas.
- Specify the following properties.

Table 1. XSLT policy properties

Property label	Required	Description	Data type
----------------	----------	-------------	-----------

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is <code>xslt</code> .	string
Description	No	A description of the policy.	string
Input	No	Indicates whether this XSLT input document uses the context current payload, or if there is no input. The check box is cleared by default, which indicates that there is no input.	boolean
Source	Yes	The XSLT transform source to execute.	string

6. Specify a version for the policy by clicking the Source icon , and completing the `version` section of the policy YAML. For example:

```
execute:
- xslt:
 version: 1.0.0
 title: xslt
 ...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

## XSLT policy examples

Examples of the OpenAPI definitions of XSLT policies.

- [Simple example with no context current payload](#)
- [Concatenation and transformation](#)
- [Obtain query parameter values and refer to context variables](#)
- [Return DataPower object name associated with a client TLS profile](#)

Attention:

- The mechanisms described here for interacting with the API Connect context are intended for v5-compatible gateways and also to provide v5 compatibility for APIs created for the API Gateway. However, if you are writing an XSLT policy or user-defined policy for a new API Gateway API, use the mechanisms described in [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway](#) for better performance and long-term support. For API Gateway APIs, the v5-compatibility functions listed here are to be used for v5 migration only.
- The XML specification <https://www.w3.org/TR/xml/> does not specify a preferred order for XML namespace (XMLNS) attributes. Best practice is to not rely upon the sequence of XMLNS attributes if you write custom parsing code.

## Simple example with no context current payload

The following is an example of where the XSLT input document does not use the context current payload (there is no input):

```
- xslt:
 title: example xslt
 source: |
 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
 <Hello>World!</Hello>
 </xsl:template>
 </xsl:stylesheet>
```

## Concatenation and transformation

The following example shows a more complex XSLT transform source, where the stylesheet concatenates two input strings and transforms the third input string to the IP address of the client:

```
- xslt:
 title: xslt
 input: true
 source: |
 <?xml version="1.0" encoding="UTF-8"?>
 <xsl:stylesheet
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:xalan="http://xml.apache.org/xslt"
 xmlns:fn="http://www.w3.org/2005/xpath-functions"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xs4xs="http://www.w3.org/2001/XMLSchema"
 xmlns:io="http://xformMessage"
 xmlns:map="http://xformMessage/xform"
 xmlns:msl="http://www.ibm.com/xmlmap"
 exclude-result-prefixes="fn dp dp map xalan msl"
 version="1.0">
 <xsl:output method="xml" encoding="UTF-8" indent="no"/>

 <!-- root wrapper template -->
 <xsl:template match="/">
 <msl:datamap>
 <xsl:choose>
 <xsl:when test="not(msl:datamap/dataObject[1]/@xsi:nil)">
 <xsl:element name="dataObject">
 <xsl:attribute name="xsi:type">
 <xsl:value-of select="'io:data'"/>
 </xsl:attribute>
```



```

 <xsl:call-template name="map:xform">
 <xsl:with-param name="data" select="msl:datamap/dataObject[1]"/>
 </xsl:call-template>
 </xsl:element>
 </xsl:when>
 <xsl:otherwise>
 <xsl:element name="dataObject">
 <xsl:attribute name="xsi:type">
 <xsl:value-of select="'io:data'"/>
 </xsl:attribute>
 <xsl:attribute name="xsi:nil">
 <xsl:text>true</xsl:text>
 </xsl:attribute>
 </xsl:element>
 </xsl:otherwise>
 </xsl:choose>
</msl:datamap>
</xsl:template>

```

```

<!-- This rule represents a type mapping: "data" to "io:data". -->
<xsl:template name="map:xform">
 <xsl:param name="data"/>
 <!-- a simple data mapping: "$data/StringOne" (string) to "StringOne" (string) -->
 <xsl:if test="$data/StringOne">
 <StringOne>
 <xsl:value-of select="concat($data/StringOne, $data/StringTwo)"/>
 </StringOne>
 </xsl:if>
 <!-- a simple mapping with no associated source: to "StringTwo" (string) -->
 <StringTwo>
 <xsl:value-of select="dp:client-ip-addr()"/>
 </StringTwo>
 <!-- a simple data mapping: "$data/NumberOne" (int) to "NumberOne" (int) -->
 <xsl:if test="$data/NumberOne">
 <NumberOne>
 <xsl:value-of select="$data/NumberOne"/>
 </NumberOne>
 </xsl:if>
 <!-- a simple data mapping: "$data/NumberTwo" (int) to "NumberTwo" (int) -->
 <xsl:if test="$data/NumberTwo">
 <NumberTwo>
 <xsl:value-of select="$data/NumberTwo"/>
 </NumberTwo>
 </xsl:if>
 <!-- a simple data mapping: "$data/NumberThree" (int) to "NumberThree" (int) -->
 <xsl:if test="$data/NumberThree">
 <NumberThree>
 <xsl:value-of select="$data/NumberThree"/>
 </NumberThree>
 </xsl:if>
</xsl:template>

```

```

<!-- ***** Utility Templates ***** -->
<!-- copy the namespace declarations from the source to the target -->
<xsl:template name="copyNamespaceDeclarations">
 <xsl:param name="root"/>
 <xsl:for-each select="$root/namespace::*[not(name() = '')]">
 <xsl:copy/>
 </xsl:for-each>
</xsl:template>
</xsl:stylesheet>

```

## Obtain query parameter values and refer to context variables

The following example shows a complete OpenAPI source file. The API includes an XSLT policy that obtains a query parameter value in XSLT, and also uses the `getvariable` method to retrieve the value of the context variable `request.headers.user-agent`.

```

swagger: '2.0'
info:
 x-ibm-name: xslt
 title: xslt
 version: 1.0.0
schemes:
 - https
host: $(catalog.host)
basePath: /xslt
consumes:
 - application/json
produces:
 - application/json
securityDefinitions:
 clientIdHeader:
 type: apiKey
 in: header
 name: X-IBM-Client-Id
security:
 - clientIdHeader: []
x-ibm-configuration:
 testable: true
 enforced: true
cors:
 enabled: true
assembly:
 execute:

```

```

- operation-switch:
 title: operation-switch
 case:
 - operations:
 - verb: get
 path: /hello
 execute:
 - xslt:
 title: SayHello
 input: false
 source: |
 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
 <xsl:template match="/">
 <xsl:element name="APIC">
 <xsl:text>Hello World!</xsl:text>
 </xsl:element>
 </xsl:template>
 </xsl:stylesheet>
 - operations:
 - verb: get
 path: /getContextQueryVar
 execute:
 - xslt:
 title: GetContextQueryVar
 input: false
 source: |
 <xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:apim="http://www.ibm.com/apimanagement">
 <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_local:isp_policy_apim.custom.xsl"/
>
 <xsl:template match="/">
 <xsl:call-template name="apim:output">
 <xsl:with-param name="mediaType" select="'application/xml'"/>
 </xsl:call-template>
 <APIC>
 <xsl:element name="apim:getVariable">
 <xsl:element name="useragent">
 <xsl:value-of select="apim:getVariable('request.headers.user-agent')"/>
 </xsl:element>
 <xsl:element name="query">
 <xsl:value-of select="apim:getVariable('request.querystring')"/>
 </xsl:element>
 </xsl:element>
 </APIC>
 </xsl:template>
 </xsl:stylesheet>
 - operations:
 - verb: get
 path: /getQuery
 execute: []
 otherwise:
 - throw: null
 title: handling unknown operation
 name: Unsupported
 catch:
 - errors:
 - Unsupported
 execute:
 - set-variable:
 actions:
 - set: message.body
 value: '<error>Not Supported</error>'
 phase: realized
paths:
/hello:
 get:
 responses:
 '200':
 description: 200 OK
/getContextQueryVar:
 get:
 responses:
 '200':
 description: 200 OK
definitions: {}
tags: []

```

## Return DataPower object name associated with a client TLS profile

The following example uses the `apim:getTLSProfileObjName` function to return the DataPower object name associated with a specified client TLS profile.

```

- xslt:
 title: xslt
 input: false
 version: 2.0.0
 source: >-
 <?xml version="1.0" encoding="UTF-8"?>
 <xsl:stylesheet
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:apim="http://www.ibm.com/apimanagement"
 exclude-result-prefixes="dp apim"

```

```

extension-element-prefixes="dp"
version="1.0">

<!-- For apigw ... store:///dp/apim.custom.xsl -->
<!-- For v5/v5c ... local:///isp/policy/apim.custom.xsl -->
<!-- <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"/
> -->
<xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl"/>

<xsl:template match="/">
 <xsl:variable name="httpHeaders">
 <header name="Accept">application/xml</header>
 </xsl:variable>
 <!--
 For v5, the name returned is client:<orgname>-my-test
 For v5c, the name returned is client:<orgname>-my-testV1.0.0
 For API Gateway, the name returned is client:<orgname>_<catalogname>_tlsp-my-testV1.0.0
 -->
 <xsl:variable name="tlsClientProfile" select="apim:getTLSProfileObjName('my-test')"/>
 <xsl:variable name="targetUrl" select="'https://127.0.0.1:6201/PingRequest'"/>
 <dp:url-open target="{&targetUrl}" response="responsecode" timeout="20" http-headers="{&httpHeaders}" ssl-proxy="
 {&tlsClientProfile}"/>
</xsl:template>

</xsl:stylesheet>

```

For more examples of how to use XSLT to access and modify properties and context, see [Implementation code examples](#) and, if you are using the DataPower® API Gateway, [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway](#).

## Implementation code examples

Example XSLT and GatewayScript code snippets.

Note: If you are using GatewayScript, you must include one or other of the following commands depending on your gateway type:

```

DataPower Gateway (v5 compatible) var apic = require('./apim.custom.js');
API Gateway var apic = require('apim');

```

where *apic* is the common name used for the GatewayScript examples in this topic. However, *apic* could be any given name of your choice, for example you could use:

```
var apim = require('./apim.custom.js');
```

and then you would start your calls with `apim`.

- [Access to input properties code snippet](#)
- [Access to runtime context code snippet](#)
- [Access to input payload code snippet](#)
- [Access to HTTP headers code snippet](#)
- [Modify the payload code snippet](#)
- [Configure error information code snippet](#)
- [Accessing the caught exception in a catch block](#)
- [Set variables code snippet](#)

## Access to input properties code snippet

The following code block shows an example of how to access the input properties by using the XSLT `policyProperties()` function. The example defines a property that is named `a_property`, which is declared as an integer value, but is retrieved in XSLT as text.

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:func="http://exslt.org/functions"
 xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

 <!-- Contains the APIM functions -->
 DataPower Gateway (v5 compatible) <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
/>
 API Gateway <xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

 <xsl:template match="/">
 <xsl:variable name="p" select="apim:policyProperties()" />
 <xsl:message>
 The value of my input property is
 <xsl:value-of select="$p/a_property" />
 </xsl:message>
 </xsl:template>

</xsl:stylesheet>

```

If you are using GatewayScript, call the following function:

```
apic.getPolicyProperty (propertyName)
```

where *propertyName* is the name of the input property that you want to access. If the input property name is blank, the action will return all input properties.

## Access to runtime context code snippet

The following code block shows an example of how to access the runtime context by using the XSLT `getContext()` function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:func="http://exslt.org/functions"
 xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

 <!-- Contains the APIM functions -->
 <xsl:import
 href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
 />

 <xsl:include
 href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

 <xsl:template match="/">
 <xsl:variable name="client-id" select="apim:getContext('client.app.id') " />
 <xsl:message>
 The calling application is
 <xsl:value-of select="$client-id" />
 </xsl:message>
 </xsl:template>

</xsl:stylesheet>
```

If you are using GatewayScript, call the following function:

```
apic.getContext (varName)
```

where *varName* is the name of the context variable that you want to access.

For a complete list of context variables, see [API Gateway context variables](#). If you are using the DataPower® API Gateway, see also [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway](#).

## Access to input payload code snippet

The following code block shows an example of how to access the input payload by using the XSLT `payloadRead()` function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:func="http://exslt.org/functions"
 xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

 <!-- Contains the APIM functions -->
 <xsl:import
 href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
 />

 <xsl:include
 href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

 <xsl:template match="/">
 <xsl:variable name="input" select="apim:payloadRead() " />
 <xsl:message>
 The input payload is
 <xsl:copy-of select="$input" />
 </xsl:message>
 </xsl:template>

</xsl:stylesheet>
```

If you are using GatewayScript, call the following function:

```
apic.readInput (callback)
```

A callback is required because the actual payload read is asynchronous. The callback method is called when the payload is ready.

This function returns an XML node-set that contains the payload of the request. If the payload is in JSON format, a JSONx node-set is returned that can then be manipulated within an XSLT or GatewayScript stylesheet. If the payload is not in JSON or XML format, the node-set that is returned is empty.

The following example shows how to use the `payloadType()` function to determine what type of payload (XML or JSONx) will be returned by the XSLT `payloadRead()` function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:func="http://exslt.org/functions"
 xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

 <!-- Contains the APIM functions -->
```

```

<xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
/>
<xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

<xsl:template match="/">
 <xsl:variable name="payloadType" select="apim:payloadType()" />
 <xsl:message>
 <xsl:text>Payload type is [</xsl:text>
 <xsl:value-of select="$payloadType" />
 <xsl:text>]</xsl:text>
 </xsl:message>
</xsl:template>

</xsl:stylesheet>

```

## Access to HTTP headers code snippet

The following code block shows an example of how to access the HTTP headers in XSLT by using the `getContext()` function.

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:func="http://exslt.org/functions"
 xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

 <!-- Contains the APIM functions -->
 <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
/>
 <xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

 <xsl:template match="/">
 <xsl:variable name="content-type" select="apim:getContext('request.headers.content-type')" />
 <xsl:message>
 The request content type is
 <xsl:value-of select="$content-type" />
 </xsl:message>
 </xsl:template>

</xsl:stylesheet>

```

If you are using GatewayScript, call the following function:

```
apic.getContext(request.headers.headerName)
```

where `headerName` maps to the name of the header you want to access.

Note: Access or modification of HTTP headers by using DataPower extensions, such as `dp:set-request-header`, is not advisable, as such actions might yield unexpected results when the policy is combined with other policies and assembly steps.

## Modify the payload code snippet

The output from a user-defined policy must be an XML node-set, which represents an XML or SOAP message, or a JSON message by using JSONx. The following code block shows an example of how to modify the payload in XSLT. To assist the API Gateway policy framework to accept the new or transformed message, call the `apim-output` template, as shown in the following example.

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:apim="http://www.ibm.com/apimanagement"
 xmlns:jsonx="http://www.ibm.com/xmlns/prod/2009/jsonx">

 <!-- Contains the APIM functions -->
 <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
/>
 <xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

 <xsl:template match="/">
 <!-- Creates a JSON document (empty object is for simplicity) -->
 <jsonx:object>
 </jsonx:object>

 <!-- Indicates the media type of the output being produced -->

 <xsl:call-template name="apim:output">
 <xsl:with-param name="mediaType" select="'application/json'" />
 </xsl:call-template>
 </xsl:template>

</xsl:stylesheet>

```

where `mediaType`:

- `'application/json'` is when the output is written in JSONx format.
- `'application/xml'` is when the output is written in XML format.

If you are using GatewayScript, call the following function:

```
apic.output(mediaType)
```

where `mediaType` is:

- `application/json` is when the output is written in JSONx format.
- `application/xml` is when the output is written in XML format.

Specifying the media type allows the next steps in the assembly flow to understand how to process the new payload.

Tip: The output from a user-defined policy must be XML or JSONx. JSONx is an IBM standard format to represent JSON as XML. One way to convert output GatewayScript JSON data into JSONx, is to add a **Convert Query Params to XML** action to follow the GatewayScript action within the same policy rule. The **Convert Query Params to**

**XML** action must have an **Input Conversion** with the **Default**

**Encoding** set to **JSON**. The output from the GatewayScript action must be the input for the **Convert Query Params to XML** action for JSONx to be produced.

## Configure error information code snippet

The following XSLT code block shows an example of how to configure the policy implementation to produce error information by calling the `apim-error` template.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:apim="http://www.ibm.com/apimanagement">

 <!-- Contains the APIM functions -->
 <xsl:import
 href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xml"
 />

 <xsl:include
 href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xml" />

 <!-- Indicates this policy has a failure and provides
 additional information for the client application -->
 <xsl:template match="/">
 <xsl:call-template name="apim:error">
 <xsl:with-param name="httpCode" select="'401'" />
 <xsl:with-param name="httpReasonPhrase" select="'Unauthorized'" />
 <xsl:with-param name="errorMessage" select="'Please select a Plan'" />
 </xsl:call-template>
 </xsl:template>
</xsl:stylesheet>
```

where:

- `httpCode` is the code of the required error message.
- `httpReasonPhrase` is the reason for the error.
- `errorMessage` is the suggested action for the user.

If you are using GatewayScript, call the following function:

```
apic.error(name, httpCode, httpReasonPhrase, message)
```

where:

- `name` is the name of the error.
- `httpCode` is the code of the required error message.
- `httpReasonPhrase` is the reason for the error.
- `message` is the suggested action for the user.

## Accessing the caught exception in a catch block

The following XSLT code block shows an example of how, in the `catch` block of an API assembly, you can obtain the details of the current caught exception. A possible use would be to create a custom error response using the details of the caught exception.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:apim="http://www.ibm.com/apimanagement"
 xmlns:apigw="http://www.ibm.com/xmlns/datapower/2017/11/apigateway"
 extension-element-prefixes="dp">
 <xsl:exclude-result-prefixes="dp apim">
 <xsl:exclude-result-prefixes="dp apim apigw">

 <xsl:output method="xml" />
```

```

<!-- Contains the APIM functions -->
<xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_local:isp_policy_apim.custom.xsl"
/>
<xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_store:dp_apim.custom.xsl" />

<xsl:template match="/">

 <xsl:variable name="exception" select="apim:getError()" />

 <xsl:variable name="exception" select="apigw:get-error()" />
 <!-- output desired error message based on the exception -->
 <myError>
 <errorReason><xsl:value-of select="$exception/error/message" /></errorReason>
 </myError>

 <!-- Propagate the HTTP status code and reason phrase from the exception -->
 <xsl:call-template name="apim:setVariable">
 <xsl:with-param name="varName" select="'message.status.code'"/>
 <xsl:with-param name="value" select="$exception/error/status/code"/>
 <xsl:with-param name="action" select="'Set'"/>
 </xsl:call-template>

 <xsl:call-template name="apim:setVariable">
 <xsl:with-param name="varName" select="'message.status.reason'"/>
 <xsl:with-param name="value" select="$exception/error/status/reason"/>
 <xsl:with-param name="action" select="'Set'"/>
 </xsl:call-template>

</xsl:template>

</xsl:stylesheet>

```

The `apim:getError()` and `apigw:get-error()` functions return an XML node set; for example:

```

<?xml version="1.0" encoding="UTF-8"?>
<error>
 <name>RuntimeException</name>
 <message>This is a thrown Runtime Error</message>
 <policyTitle>Throw Runtime Error</policyTitle>
 <status>
 <code>500</code>
 <reason>Internal Server Error</reason>
 </status>
</error>

```

If you are using GatewayScript, call the following function:

```
apim.getError()
```

which returns a JSON object; for example:

```

{
 "name": "OperationError",
 "message": "This is a thrown Operation Error",
 "policyTitle": "Throw Operation Error",
 "status": {
 "code": "500",
 "reason": "Internal Server Error"
 }
}

```

## Set variables code snippet

The following XSLT code block shows an example of how to set a runtime variable to a specified string value by calling the `setVariable` template.

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:func="http://exslt.org/functions"
 xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

 <!-- Contains the APIM functions -->
 <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_local:isp_policy_apim.custom.xsl"
/>
 <xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_store:dp_apim.custom.xsl" />

 <xsl:template match="/">
 <xsl:call-template name="apim:setVariable">
 <xsl:with-param name="varName" select="'serviceEndpoint'"/>
 <xsl:with-param name="value" select="'https://endpoint.host.com/data'"/>
 </xsl:call-template>
 <xsl:message>
 <xsl:text>Variable [</xsl:text>
 <xsl:value-of select="'serviceEndpoint'"/>
 <xsl:text>] set to [</xsl:text>

```

```

 <xsl:value-of select="'https://endpoint.host.com/data'" />
 <xsl:text>]</xsl:text>
 </xsl:message>
</xsl:template>

```

```
</xsl:stylesheet>
```

where:

- **varName** is the name of the runtime variable that you want to set a value to.
- **value** is the string value that you want to set the variable to. This can be a literal value, or another variable. For example, to set your named variable to the value of the Content-Type header in a request, you would specify the **value** as `$(request.headers.content-type)`.

If you are using GatewayScript, call the following function:

```
apic.setvariable(varName, varValue, action)
```

where:

- **varName** is the name of the runtime variable that you want to set a value to, or that you want to add or clear.
- **varValue** is the string value that you want to set the variable to. This can be a literal value, or another variable. For example, to set your named variable to the value of the Content-Type header in a request, you would specify the **varValue** as `request.headers.content-type`. This property is only required when **set** or **add** is specified as the action.
- **action** is the action that you want to apply to the variable. Valid options are:
  - **set**
  - **add**
  - **clear**

If no option is set, the default option of **set** is applied.

The following XSLT example shows how to retrieve the value of a runtime variable by using the `getVariable()` function.

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:func="http://exslt.org/functions"
 xmlns:apim="http://www.ibm.com/apimanagerment" extension-element-prefixes="dp func apim">

 <!-- Contains the APIM functions -->
 <xsl:import
 href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
 />

 <xsl:include
 href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

 <xsl:template match="/">
 <xsl:variable name="varValue" select="apim:getVariable('serviceEndpoint') " />
 <xsl:message>
 <xsl:text>Variable [
 <xsl:value-of select="'serviceEndpoint'" />
 <xsl:text>] = [
 <xsl:value-of select="$varValue" />
 <xsl:text>]</xsl:text>
 </xsl:message>
 </xsl:template>
</xsl:stylesheet>

```

where

- **varValue** is the name of the runtime variable that you want to retrieve a value for.

If you are using GatewayScript, call the following function:

```
apic.getvariable(varName)
```

where **varName** is the name of the runtime variable that you want to retrieve a value for.

## Logic Constructs

IBM® API Connect includes a number of logic constructs that you can use to apply preconfigured logic to an assembly to control the flow of data through your assembly when the API is called.

Logic constructs are configured in the context of an API. You can use the API Designer assembly editor to add a logic construct to an API and to configure the properties for that construct.

The logic constructs are described in the following subtopics:

- **if**  
Use the if construct to apply a section of the assembly when a condition is fulfilled.
- **operation-switch**  
Use the operation-switch component to apply a section of the assembly to a specific operation.
- **switch**  
Use the switch component to execute one of a number of sections of the assembly based on which specified condition is fulfilled.



- [throw](#)  
Use the throw policy to throw an error when it is reached during the execution of an assembly flow.

## Related concepts

- [The assembly editor](#)

## if

Use the if construct to apply a section of the assembly when a condition is fulfilled.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway, functionality provided by <a href="#">switch</a>	

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [if](#).

An if construct provides a way to branch an API's assembly when a specified condition is fulfilled. Each if construct contains a section of the assembly that is only executed when the script within the construct returns a **true** value.

When using the assemble view's property sheet, use the Condition field to write your condition that returns **true** or **false**.

If you want one or more policies or constructs to be executed when the condition of the if construct is fulfilled, drag the new policy or construct onto one of the dashed boxes that are displayed within the if construct. Constructs and policies included in the if construct are part of the case that is executed when the condition of the if construct is returned as true.

For information about the OpenAPI implementation of an if construct, see [if](#).

In the Condition field, use the form `apim.getvariable('context.location.variable')` to reference your variables, where *context* is the context that you want to reference, *location* is the location of the variable within that context, and *variable* is the name of the variable.

## Construct property details

You can configure a construct's properties in the property sheet in the assemble view.

Table 2. The properties of an if construct

Property	Required	Description
Title	No	A custom title for your construct when it is displayed in the canvas. If a title is not specified, <b>if</b> is used by default.
Description	No	A description of your construct, it is not displayed on the canvas.
Condition	Yes	Use GatewayScript to provide conditions. A list of context variables that you can use to generate conditions can be found in <a href="#">API Connect context variables</a> .

## operation-switch

Use the operation-switch component to apply a section of the assembly to a specific operation.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [operation-switch](#).

An operation-switch component provides a way to branch an API's assembly depending on the operation that is called. For each case, a separate piece of the assembly is applied to the operations that belong to the case.

When using the assemble view's property sheet, click the Case field to view suggested operations. Type in the Case field to refine the list of suggested operations.

Each case behaves as an assembly. To add components to a case, drag the component from the palette to the area included in the operation-switch component. Dashed boxes are displayed where the component can be included in the case.

For information about the OpenAPI implementation of an operation-switch component, see [operation-switch](#).

Restriction: Nesting an operation-switch component inside an if or switch construct, or another operation-switch component, is **not** supported.

## Component property details

You can configure a component's properties in the property sheet in the assemble view.

Table 2. The properties of an operation-switch component

Property	Required	Description
Title	No	A custom title for your component when it is displayed in the canvas. If a title is not specified, <code>operation-switch</code> is used by default.
Description	No	A description of your component, it is not displayed on the canvas.
Case	Yes	Each case includes one or more operations to which the branch provided by the operation switch applies.

## switch

Use the switch component to execute one of a number of sections of the assembly based on which specified condition is fulfilled.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [switch](#).

A switch construct provides a way to branch an assembly based on multiple conditions. Each switch component contains multiple cases, each corresponding to a section of the assembly that is only executed when the condition or operation specified by the case is met or used. Additionally, an otherwise case executes when no other case is fulfilled.


Add new cases by clicking + Case and add an "otherwise" case by clicking + Otherwise.

If multiple cases are fulfilled, the highest priority case will be executed. Change the priority of cases by clicking the Move up ↑ and Move down ↓ icons.

To configure a case to be executed if a specific operation is called, use the search operations field and select your operation from the list. You can refine the results of the search by typing in the search operations field.

To configure a case to be executed based on a GatewayScript condition, click edit condition and enter your script in the "Condition editor" window. When you have supplied a script, you can edit your script either in the Condition field or by clicking edit condition.

If you are using the DataPower Gateway (v5 compatible), you enter your condition as GatewayScript directly in a code area in the Condition editor. If you are using DataPower API Gateway, the Condition editor provides a script builder to help you construct your condition script; for more information, see [Using the switch policy condition editor](#).

To delete a case, click the Remove case icon .


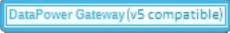
If you want one or more policies or constructs to be executed when the condition of a case is fulfilled, drag the new policy or construct onto one of the dashed boxes that are displayed within the case's section of the switch construct.

Note: A switch case **must** contain at least one policy, otherwise the Gateway server returns an error.

## Construct property details

You can configure a construct's properties in the property sheet in the assemble view.

Table 2. The properties of a switch construct

Property	Required	Description
Title	No	A custom title for your construct when it is displayed in the canvas. If a title is not specified, <code>switch</code> is used by default.
Description	No	A description of your construct, it is not displayed on the canvas.
case	Yes (one or more)	Specify one or more operations or write a script for a condition.  Use the Condition editor to construct your condition script. See <a href="#">Using the switch policy condition editor</a> .  Use GatewayScript to provide conditions. You can reference variables by using the form <code>apim.getvariable('context.location.variable')</code> , where <code>context</code> is the context that you want to reference, <code>location</code> is the location of the variable within that context, and <code>variable</code> is the name of the variable. A list of context variables that you can use to generate conditions can be found in <a href="#">API Connect context variables</a> .
otherwise	No	Add an otherwise case if you want to execute a section of the assembly when no other cases are fulfilled.

- [Using the switch policy condition editor](#)

The switch policy condition editor provides a user interface to help you build the conditions for the `case` clauses in a switch policy.



## Using the switch policy condition editor

The switch policy condition editor provides a user interface to help you build the conditions for the **case** clauses in a switch policy.

You build a condition script by using the Add Group and Add Condition options, together with operator selections, and pre-supplied function selections. The Output field shows the resultant condition script, and updates dynamically as you build your condition.

## Adding conditions

When you open the policy condition editor, an initial condition is pre-supplied ready for you to configure.

Use the drop-down list to select the function that you want to begin your condition.

The following JSONata functions are supported:

- Aggregation functions:
  - `$average(array)`
  - `$max(array)`
  - `$min(array)`
  - `$sum(array)`
- Array functions:
  - `$append(array1, array2)`
  - `$count(array)`
  - `$reverse(array)`
  - `$sort(array [, function])`
  - `$zip(array1, ...)`
- Boolean functions:
  - `$boolean(arg)`
  - `$exists(arg)`
  - `$not(arg)`
- Numeric functions:
  - `$abs(number)`
  - `$ceil(number)`
  - `$floor(number)`
  - `$formatBase(number [, radix])`
  - `$number(arg)`
  - `$power(base, exponent)`
  - `$round(number [, precision])`
  - `$sqrt(number)`
- Object functions:
  - `$keys(object)`
  - `$lookup(object, key)`
  - `$merge(array<object>)`
  - `$spread(object)`
  - `$type(value)`  
In addition to the standard values returned by `$type()`, the API gateway implementation of `$type()` can return the following values: **binary**, **empty**, **graphql**, **stream**, or **xml**.
- String functions:
  - `$contains(str, pattern)`
  - `$join(array[, separator])`
  - `$length(str)`
  - `$lowercase(str)`
  - `$match(str, pattern [, limit])`
  - `$pad(str, width [, char])`
  - `$replace(str, pattern, replacement [, limit])`
  - `$split(str, separator [, limit])`
  - `$string(arg)`
  - `$substring(str, start[, length])`
  - `$substringAfter(str, chars)`
  - `$substringBefore(str, chars)`
  - `$trim(str)`
  - `$uppercase(str)`
- You can use the following functional extensions to standard JSONata notation. Each extension corresponds to a part of the API context.

Table 1. Functional extensions to JSONata

Extension	Variable	Description
<code>\$header(name)</code>	<code>message.headers.name</code>	Message header
<code>\$httpVerb()</code>	<code>request.verb</code>	HTTP method of the request
<code>\$operationID()</code>	<code>api.operation.id</code>	ID of the operation
<code>\$operationPath()</code>	<code>api.operation.path</code>	Path of the operation

Extension	Variable	Description
<code>\$queryParameter('name')</code>	<ul style="list-style-type: none"> <li><code>request.parameters.name.locations</code> The supported keyword is <code>query</code>.</li> <li><code>request.parameters.name.values</code></li> </ul>	Searches for the index of <code>query</code> in <code>request.parameters.name.locations</code> and returns <code>request.parameters.name.values[index]</code> , where <code>[index]</code> is the value for <code>query</code> in locations. Parameter values are not URL decoded.
<code>\$statusCode()</code>	<code>message.status.code</code>	Status code
<code>\$storageType([arg])</code>	<code>variable.body</code> You can specify any variable in the API context. When no variable is specified, the default variable <code>message.body</code> is used.	Storage type of the message. The supported values are <code>binary</code> , <code>empty</code> , <code>graphql</code> , <code>json</code> , <code>stream</code> , or <code>xml</code> .
<code>\$urlParameter('name')</code>	<ul style="list-style-type: none"> <li><code>request.parameters.name.locations</code> The supported keywords are <code>path</code> and <code>query</code></li> <li><code>request.parameters.name.values</code></li> </ul>	Searches for the index of <code>path</code> and <code>query</code> in <code>request.parameters.name.locations</code> and returns a single array that contains both <code>path</code> and <code>query</code> values from <code>request.parameters.name.values</code> . When the URL contains both path and query parameter values, the array includes the path values first followed by the query values. The values of each parameter type are added in the order that they are received. Parameter values are URL decoded. For example, the following URL contains both path and query parameter values. <code>http://example.com/petstore/cats/adopt?breed=Sphynx&amp;breed=Siamese</code> The <code>\$urlParameter('breed')</code> URL returns the following array of values. <code>[cats, adopt, Sphynx, Siamese]</code> In this example, the URL includes an API path that is configured as <code>/petstore/{breed}/{breed}</code> , where <code>breed</code> is configured to be a path parameter of the API path. As a result, <code>cats</code> and <code>adopt</code> are included in the output.
<code>\$xpath(path, xpathExpression)</code>	You can specify any writable variable in the API context. The <code>xpathExpression</code> must be a literal string.	Allows use of XPath expressions. The following example specifies all price elements in the source. <code>\$xpath(\$, '//price')</code>

Table 2. Functional extensions to JSONata for GraphQL

Extension	Variable	Description
<code>\$gqlActiveOperation([graphql_message])</code>	<code>message.body</code>	Gets the active operation found in the specified GraphQL message. The <code>operationName</code> must be the same as the name of the active operation.
<code>\$gqlAlias(graphql_field_node)</code>	<code>message.body</code>	Gets the alias of a GraphQL field node.
<code>\$gqlFragments([graphql_message])</code>	<code>message.body</code>	Gets the fragments found in the specified GraphQL message.
<code>\$gqlName([graphql_node])</code>	<code>message.body</code>	Gets the node name. For operations, the node name is the <code>operationName</code> . For fields, fragment definitions, arguments, and other elements, the node name is the name of the element. By default, the <code>operationName</code> of <code>message.body</code> is retrieved.
<code>\$gqlOperations([graphql_message])</code>	<code>message.body</code>	Gets the operations found in the specified GraphQL message.
<code>\$gqlType([graphql_node])</code>	<code>message.body</code>	The operation type of the active operation is retrieved for Query, Mutation, and Subscription query types.

- You can use the `&` (concatenation) navigation operator.

You can use the following JSONata numeric operators:

- `+` (addition)
- `-` (subtraction)
- `*` (multiplication)
- `/` (division)
- `%` (modulo)

You can use the following JSONata comparison operators for number values or strings:

- `=`
- `!=`
- `<`
- `>`
- `<=`
- `>=`

You can also use the following operators and expressions.

- Parentheses to convert a sequence into an array, specify operator precedence, or compute complex expressions on a context value.
- Array ranges and predicate expressions.
- Single asterisk (`*`) and double asterisk (`**`) wildcard characters.

The following elements of a GraphQL query can be exposed in JSONata notation using the syntax shown.

- `query`  
The entire GraphQL query including operations and fragments.

- **operationName**  
For an anonymous operation, **operationName** can be empty.
- **~fragmentSpreadName**
- **on~typeCondition**
- **~~fragmentDefinitionName**

You can also use the following functional extensions to standard JSONata notation. Each extension corresponds to a part of the API context.

Extension	Variable	Description
<code>\$header (name)</code>	<code>message.headers.name</code>	Message header
<code>\$httpVerb ()</code>	<code>request.verb</code>	HTTP method of the request
<code>\$operationID ()</code>	<code>api.operation.id</code>	ID of the operation
<code>\$operationPath ()</code>	<code>api.operation.path</code>	Path of the operation
<code>\$queryParameter ('name')</code>	<ul style="list-style-type: none"> <li>• <code>request.parameters.name.locations</code></li> <li>• <code>request.parameters.name.values</code></li> </ul>	Returns the value of a request query parameter given the parameter name. The function searches the <code>request.parameters.name.locations</code> array for the index position of the value <code>query</code> , and returns <code>request.parameters.name.values[index]</code> , where <code>index</code> is the index position. Parameter values are not URL decoded.
<code>\$statusCode ()</code>	<code>message.status.code</code>	Status code
<code>\$storageType ([arg])</code>	<b>variable.body</b> You can specify any variable in the API context. When no variable is specified, the default variable <code>message.body</code> is used.	Storage type of the message. The supported values are: <ul style="list-style-type: none"> <li>• <code>binary</code></li> <li>• <code>json</code></li> <li>• <code>stream</code></li> <li>• <code>xml</code></li> </ul>
<code>\$urlParameter ('name')</code>	<ul style="list-style-type: none"> <li>• <code>request.parameters.name.locations</code></li> <li>• <code>request.parameters.name.values</code></li> </ul>	Given a request parameter name, returns the parameter values for all occurrences of that parameter as a path parameter or query parameter. The function searches the <code>request.parameters.name.locations</code> array for the index positions of the values <code>path</code> and <code>query</code> , and returns, in a single array, the <code>request.parameters.name.values[index]</code> values for all identified index positions. When the URL contains both path and query parameter values, the array includes the path values first followed by the query values. The values of each parameter type are added in the order that they are received. Parameter values are URL decoded. <p>For example, the following URL contains both path and query parameter values:</p> <pre>http://example.com/petstore/cats/adopt?breed=Sphynx&amp;breed=Siamese</pre> <p>The function call <code>\$urlParameter ('breed')</code> returns the following array of values:</p> <pre>[cats, adopt, Sphynx, Siamese]</pre> <p>In this example, the URL includes an API path that is configured as <code>/petstore/{breed}/{breed}</code>, where <code>breed</code> is configured to be a path parameter of the API path. As a result, <code>cats</code> and <code>adopt</code> are included in the output.</p>
<code>\$xpath (path, xpathExpression)</code>	You can specify any writable variable in the API context. The <code>xpathExpression</code> must be a literal string.	Allows use of XPath expressions

After you select the function, additional fields are displayed, depending on the selected function, so that you can complete the condition. For example, if you select the function `$httpVerb ()`, a comparison operator selection list is displayed (= or !=), together with a list of HTTP verbs to select on the right hand side of the expression (`GET`, `PUT`, `POST`, `DELETE`, `HEAD`, `OPTIONS`, or `PATCH`).

If you select NOT, your condition is negated with a `$not` function.

To add further conditions, click Add Condition; you then select whether to insert an `and` or an `or` operator between this condition and the preceding one.

To group two or more conditions inside parentheses, click Add Group, then add your conditions to the group; you then select whether to insert an `and` or an `or` operator between this group and the preceding condition or group.

To write your own condition from scratch, select Custom, then enter your script in the field provided.

## Example

To build the following condition:

```
($statusCode () != 200 and ($httpVerb () = 'GET' or $httpVerb () = 'PUT'))
```

complete these steps in the condition editor:

1. Select `$statusCode ()`, select != for the comparison operator, and enter `200` in the field on the right hand side of the condition.
2. Click Add Group.
3. In the new group sub pane that is displayed, click Add Condition.
4. Select `httpVerb ()`, leave the comparison operator as =, and select `GET` on the right hand side of the condition.
5. Click Add Condition in the group sub pane again.
6. Select `httpVerb ()`, leave the comparison operator as =, and select `PUT` on the right hand side of the condition.
7. For the comparison operator between the two conditions, select `or`.

## Simple condition statements

The following examples show conditions that use a single function.

This example uses the `$httpVerb()` extension to specify the HTTP method of the request.

```
$httpVerb()="GET"
```

This example uses the `$operationPath()` extension to specify the path of the operation.

```
$operationPath()="/base/path-2"
```

This example uses the `$operationID()` extension to specify the operation ID.

```
$operationID()="test-gatewayscript-GET"
```

This example uses the `$statusCode()` extension to specify the message status code.

```
$statusCode()=200
```

This example uses the `$header(name)` extension to specify the content type of the message header.

```
$header("Content-Type")="application/json"
```

## Combining conditions with logical operators

You can use `and` and `or` operators to combine multiple functions in a single condition.

This example specifies an HTTP GET request and an API operation path equal to `test-gatewayscript-GET`.

```
$httpVerb()="GET" and $operationPath()="test-gatewayscript-GET"
```

This example specifies either a `POST` or `PUT` request.

```
$httpVerb()="POST" or $httpVerb()="PUT"
```

This example specifies an API operation ID equal to `test-gatewayscript_POST` and a message status code equal to `200`, or an API operation ID equal to `test-gatewayscript-GET` and a message status code equal to `500`.

```
($operationID()="test-gatewayscript-POST" and $statusCode()=200) or ($operationID()="test-gatewayscript-GET" and $statusCode()=500)
```

This example specifies an API operation ID equal to `test-gatewayscript-POST` and a message status code equal to `200` with an API operation path equal to `/base/path-2`.

```
($operationID()="test-gatewayscript-POST" and $statusCode()=200) and $operationPath()="/base/path-2"
```

This example specifies a message header content type equal to `text/plain` and a message header length equal to `300`, or a message status code equal to `200`.

```
($header("Content-Type")="text/plain" and $header("Content-Length")=300) or $statusCode()= 200
```

## throw

Use the throw policy to throw an error when it is reached during the execution of an assembly flow.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.3.0 or later)

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see [throw](#).

When the throw policy is encountered, the specified error and error message is produced.

If a catch has been configured that the error produced by the throw policy fulfills, the catch will be triggered.

If no catch is triggered by the thrown error, then a `500 Internal Server Error` is returned to the API caller.

## Component property details

You can configure a component's properties in the property sheet in the assemble view.

Table 2. The properties of a throw component

Property	Required	Description
Title	No	A custom title for your component when it is displayed in the canvas. If a title is not specified, <b>throw</b> is used by default.
Error name	Yes	The error name that is thrown by the policy.

Property	Required	Description
<b>API Gateway only</b> Error status code (policy version 2.1.0 and later)	No	Specify the HTTP status code for the error. You can use the <code>\$(variable)</code> format to reference the <code>message.status.code</code> API context variable.
<b>API Gateway only</b> Error status reason (policy version 2.1.0 and later)	No	Specify the HTTP reason phrase for the error. You can use the <code>\$(variable)</code> format to reference the <code>message.status.reason</code> API context variable.
Error message	No	The error message that is returned with the error name.
<b>API Gateway only</b>		

## Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway

In the code for GatewayScript and XSLT policies in your API assemblies, you can use API context variables to work with the messages that are generated when an API is called.

All API context variables are stored in a JSON-like tree; each context variable is a node in this tree. A non-leaf node can be either an object or an array, while a leaf node can be a number, string, boolean, or document; a document can contain an XML document, BLOB, Stream, or another JSON document.

Each context variable has a name and a value. You can access a context variable by name when using the GatewayScript and XSLT APIs. Dot notation is used to access a context variable, reflecting the position of the corresponding node in the JSON-like tree; for example, `request.headers.Host`. Where a node name contains special characters, bracket notation is supported; for example, `request.headers['Content-Type']`.

For the GatewayScript policy, all the APIs for the DataPower® API Gateways are in the `context` module, which is a global module. You can therefore access the `context` module directly, you do not need to use the `require()` function first.

For the XSLT policy, the APIs are in the `http://www.ibm.com/xmlns/datapower/2017/11/apigateway` namespace; to use these APIs you must declare this namespace.

Attention: The mechanisms described here for interacting with the API Connect context are preferred and provide the best performance if you are writing new GatewayScript or XSLT policies, or if you are authoring user-defined policies. However, for v5-compatible gateways and v5 compatibility for APIs created for the API Gateway, the preferred mechanisms are as described in [GatewayScript code examples](#) and [XSLT policy examples](#).

- [Accessing context variables](#)
- [About payloads](#)
- [Manipulating the message headers](#)
- [Manipulating the message status - examples](#)
- [Accessing the message body](#)
- [Rejecting the session](#)

## Accessing context variables

### GatewayScript examples

To read the `client.app.id` context variable:

```
var clientID = context.get('client.app.id');
```

To set the value of a context variable:

```
context.set('my.vars.amount', 100);
```

The resulting tree has the following structure:

```
{ "my": { "vars": { "amount": 100 } } }
```

To delete the node `my.vars`, and all its child nodes:

```
context.clear('my.vars');
```

The resulting tree has the following structure:

```
{ "my": { } }
```

### XSLT examples

To read the `client.app.id` context variable:

```
<xsl:variable name="clientID" select="apigw:get-variable('client.app.id')"/>
```

To set the value of a context variable:

```
<apigw:set-variable name="'my.vars.amount'" value="100"/>
```

The resulting tree has the following structure:

```
{ "my": { "vars": { "amount": 100 } } }
```

To delete the node `my.vars`, and all its child nodes:

```
<apigw:clear-variable name="'my.vars'"/>
```

The resulting tree has the following structure:

```
{ "my": { } }
```

## About payloads

When the assembly flow for an API call begins, the following two message objects are predefined in the API context tree:

- **request**: holds the information for the original payload; the **request** message is a read-only object.
- **message**: holds the information for the current payload.

Typically, a message object has **headers** and **body** properties. For the current payload, the **message** message object might have **variables** and **status** properties, and other properties created during the execution of the assembly.

The following example shows the general structure of a typical message:

```
{
 "headers": {
 "Content-Type": "application/xml",
 "X-Client-ID": "1234567890"
 },
 "variables": {
 "amount": 123,
 "name": "xyz"
 },
 "status": {
 "code": 200,
 "reason": "OK"
 },
 "body": <XML/Blob/Stream/JSON Document>
}
```

## Manipulating the message headers

GatewayScript examples

To delete a header:

```
context.message.header.remove('X-My-Header');
```

To read a header:

```
var reqType = context.request.header.get('Content-Type');
var currType = context.message.header.get('Content-Type');
```

To set a header:

```
context.message.header.set('Content-Type', 'application/json');
```

To iterate through the headers:

```
var reqHdrs = context.request.headers;
for (var h in reqHdrs) {
 console.log('>>> Reading the request header: ', h, '=', reqHdrs[h]);
}

var currHdrs = context.message.headers;
for (var h in currHdrs) {
 console.log('>>> Reading the current header: ', h, '=', currHdrs[h]);
}
```

To create a message context called **headersContext**, if one doesn't already exist, and add to it a boolean context variable called **HelloWorld**, with the value **true**:

```
var ctx = context.getMessage("headersContext") || context.createMessage("headersContext");
ctx.setVariable('HelloWorld', true);
```

The **setVariable** call writes the variable into **headersContext.variables>HelloWorld**; therefore a condition statement in any subsequent **switch** policy should check for **headersContext.variables>HelloWorld** rather than **headersContext>HelloWorld**.

To read the context variable in any subsequent **gatewayscript** policy, use the following call:

```
ctx.getVariable("HelloWorld")
```

Note: If you are either migrating an API from IBM® API Connect Version 5.0, or from the DataPower Gateway (v5 compatible), to the DataPower API Gateway, or you are configuring a new API for the DataPower API Gateway, creating a message context, rather than using the context variable mechanism described in [Accessing context variables](#), is the preferred option. In the case of migration, it most closely matches the behavior of IBM API Connect Version 5.0, and **setVariable** and **getVariable** calls will continue to work without modification.

Creating a message context is also the preferred option if you are creating a new API for the DataPower API Gateway because it most closely aligns the API context with the way that various native DataPower API Gateway policies process it.

XSLT examples

To delete a header:

```
<apigw:clear-header message="message" name="X-My-Header" />
```

To read a header:

```
<xsl:variable name="reqType" select="apigw:get-header('request', 'Content-Type')"/>
<xsl:variable name="currType" select="apigw:get-header('message', 'Content-Type')"/>
```

To set a header:

```
<apigw:set-header name="Content-Type" value="text/xml"/>
```

To get all headers:



```
<xsl:variable name="allReqHdrs" select="apigw:all-headers('request')"/>
<xsl:variable name="allCurrHdrs" select="apigw:all-headers('message')"/>
```

## Manipulating the message status - examples

---

Note: Only the `message` object has the status property, not the `request` object.

GatewayScript examples

To read the status code and reason phrase:

```
var code = context.message.statusCode;
var reason = context.message.reasonPhrase;
```

To update the status code:

```
context.message.statusCode = 400; //integer, update the code, with default reason
```

To update the status code and reason phrase in a single operation:

```
context.message.statusCode = '200 Wonderful'; //string, update code along with customized reason
```

Note: The `context.message.reasonPhrase` context variable is read-only.

XSLT examples

To read the status code and reason phrase:

```
<xsl:variable name="code" select="apigw:get-variable('message.status.code')"/>
<xsl:variable name="reason" select="apigw:get-variable('message.status.reason')"/>

<apigw:set-variable name="message.status.code" value="200"/>
<apigw:set-variable name="message.status.reason" value="OK"/>
```

## Accessing the message body

---

Note: The request body is read-only.

GatewayScript examples

To read the request body:

```
context.request.body.readAsJSON(function(error, json) {});
context.request.body.readAsXML(function(error, xml) {});
context.request.body.readAsBuffer(function(error, buffer) {});
```

To read the message body:

```
context.message.body.readAsJSON(function(error, json) {});
context.message.body.readAsXML(function(error, xml) {});
context.message.body.readAsBuffer(function(error, buffer) {});
```

To write the message body and update the content type accordingly:

```
// text response
context.message.body.write("Hello world.");
context.message.header.set('Content-Type', "text/plain");

// JSON response
var pet = { "kind": "dog", "age": 3, "color": "black" };
context.message.body.write(pet);
context.message.header.set('Content-Type', "application/json");

// XML response
var resp = "<order><id>100</id><timestamp>20181231</timestamp><status>fulfilled</status></order>";
context.message.body.write(resp);
context.message.header.set('Content-Type', "text/xml");
```

XSLT examples

Note:

- There is no XSLT API to write the message body; the output of the stylesheet transformation is used to update `message.body`.
- To read the request body or the message body, you must parse `message.body` with a [Parse](#) policy before the XSLT policy is executed. At the beginning of the assembly flow, `message.body` contains `request.body` so, after parsing `message.body`, you can then read the request from `message.body`.
- If the XSLT input document uses the context current payload, `message.body` is used for the input document `'/'` that can be accessed from the XSLT policy if `message.body` has been parsed as XML in advance; this is determined by the `input` property of the XSLT policy, see [xslt](#).

Read the message body:

```
<xsl:variable name="currentPayload" select="apigw:read-payload('message')"/>
```

Read the message body when the `input` property of the XSLT policy is set to `true`:

```
<xsl:variable name="currentPayload" select="/" />
```

## Rejecting the session

---

Note: After the session is rejected, the API assembly flow is stopped and a fault response is returned, unless a catch handler is defined.

GatewayScript example

To reject the session with an error name and message, and update the status reason and code:

```
context.reject('CustomError', 'You are not authorized to make this API call');
context.message.statusCode = '401 Unauthorized';
```

XSLT example

To reject the session with an error name and message, and update the status reason and code:

```
<apigw:reject identifier="CustomError" status-code="401" reason="Unauthorized">You are not authorized to make this API call</apigw:reject>
```

- [Manipulating attachments in a message object](#)

The message object provides APIs and properties to manipulate message attachments.

API Gateway only

---

## Manipulating attachments in a message object

The message object provides APIs and properties to manipulate message attachments.

If the assembly includes a parse policy, attachments in the request or response of a multipart message can be manipulated and sent back to the client. By default, the response is sent to the client in multipart format. Alternatively, you can clear `context.message.attachments` to discard the attachments and send only `message.body` to the client.

Note: If the assembly does not include a parse policy, the following processing rules apply.

- For a multipart request where the `message.body` variable is not changed, the invoke policy keeps and passes the attachments to the server.
- For a multipart request or response where the `message.body` variable is not changed, the result API action passes the attachments as the final response.
- If the `message.body` variable is changed, attachments are discarded.
- [Retrieving an attachment](#)
- [Appending an attachment to the message](#)
- [Retrieving the number of attachments in an array](#)
- [Removing an attachment from the message](#)
- [Reading, accessing, and manipulating the body of an attachment](#)
- [Manipulating the headers of an attachment](#)

---

## Retrieving an attachment

GatewayScript examples

Retrieve the attachment by index location.

```
context.message.attachments[1]
```

Retrieve the attachment by looking up the value of `pic1` in the `Content-Id` header.

```
context.message.attachments['cid:pic1']
```

Retrieve the attachment by looking up the value of `uri1` in the `Content-Location` header.

```
context.message.attachments['uri1']
```

---

## Appending an attachment to the message

GatewayScript example

```
var headers = JSON.parse ('{ "Content-Type": "text/xml; charset=utf-8", "Content-Id": "<data>" }');
message.attachments.append(headers, 'Hello world.');
```

---

## Retrieving the number of attachments in an array

Example

```
for (var i = 0; i < context.message.attachments.count; i++) {
 context.message.attachments[i].header.set('X-Foo', 'bar');
}
```

---

## Removing an attachment from the message

GatewayScript examples

Remove the attachment by index location.

```
context.message.attachments.remove(1)
```

Remove the attachment by looking up the value of `pic1` in the `Content-Id` header.

```
context.message.attachments.remove('cid:pic1')
```

Remove the attachment by looking up the value of `uri1` in the `Content-Location` header.

```
context.message.attachments.remove('uri1')
```

---

## Reading, accessing, and manipulating the body of an attachment

#### GatewayScript examples

To read the payload of an attachment in binary and return the contents as either a **Buffer**, **Buffers**, **JSON**, or **XML NodeList** object:

```
context.message.attachments[0].body.readAsBuffer(function(errorObject,bufferObject){})
context.message.attachments[0].body.readAsBuffers(function(errorObject,buffersObject){})
context.message.attachments[0].body.readAsJSON(function(errorObject,jsonObject){})
context.message.attachments[0].body.readAsXML(function(errorObject,nodeList){})
```

To write content **Hello world** to the current payload of the attachment:

```
message.attachments[0].body.write("Hello world")
```

---

## Manipulating the headers of an attachment

#### GatewayScript examples

To retrieve the values of the whole attachment headers:

```
context.message.attachments[0].headers
```

To retrieve the value of the named header of the attachment:

```
context.message.attachments[0].header.get();
```

To delete the current **Content-Type** header of the attachment.

```
context.message.attachments[0].header.remove('Content-Type');
```

You can set the value of the named header of an attachment. The header can be coalesced and noncoalesced. Take the following code as an example.

```
context.message.attachments[0].header.set('MyCookie', ['cookie1', 'cookie2']);
```

If the value type is array, the **MyCookie** header is noncoalesced and it is present with different entries in HTTP header.

```
MyCookie: cookie1
MyCookie: cookie2
```

---

## User-defined policies

Make a user-defined policy available to a Catalog, and apply that policy to a REST or SOAP API.

A policy is a piece of configuration that controls a specific aspect of processing in the Gateway server during the handling of an API invocation at run time. IBM® API Connect provides built-in policies for many capabilities, including logging, redaction, proxy, and transformations, but you can also create user-defined policies to provide more processing control.

The following information outlines how to create and manage user-defined policies:

#### Creating a user-defined policy

A user-defined policy is created outside of API Connect by using a standard development tool, but the policy must conform to a particular schema. For information about how to create a policy, see [Authoring policies](#).

Note: If you are running your policies on a DataPower® Gateway, you must ensure that the DataPower processing to be controlled by the user-defined policy, is supported by the level of IBM DataPower Gateway that the policy will be run on. When using the GatewayScript actions, the minimum DataPower firmware that should be used is version 7.2.0.

#### Importing a user-defined policy into API Connect

To make a user-defined policy available to an API, the policy must be imported into one or more Catalogs in the API Manager. If you also want to be able to apply the policy to your policy assembly in API Designer, you will have to import that policy into the developer toolkit environment.

Make your policy accessible to API Designer by using the developer toolkit. For detailed instructions, see [Packaging and importing your policies into IBM API Connect](#).

Make your policy available by using the API Manager UI. For detailed instructions, see:

- [Importing a user-defined policy into a Catalog](#). If you want to create a new Catalog and import a user-defined policy as part of the configuration, follow the instructions in [Creating and configuring Catalogs](#).
- [Packaging and importing your policies into IBM API Connect](#).

Note: You must import a user-defined policy into every Catalog in API Manager that you require the policy to run in.

#### Applying a user-defined policy to an API

Apply a user-defined policy to an API definition by using the assemble view in API Designer or API Manager, to add the component to your assembly and configure its properties. For detailed instructions, see [Including elements in your OpenAPI 2.0 API assembly](#), [Including elements in your OpenAPI 2.0 API assembly](#).

#### Deleting a user-defined policy

Delete a user-defined policy from a Catalog, by clicking the Delete icon next to the policy in the associated Catalog in API Manager.

---

## Tags in API Connect

There are a number of forms of tagging in IBM® API Connect.

- You can create tags in an API that are visible in its OpenAPI definition and can be used to filter operations in the Developer Portal.
- You can tag items in the Developer Portal so that they can be searched for.

## Tags within an API

The OpenAPI specification allows the inclusion of tags in an API definition, and you can create these tags through the API Manager user interface. You can assign tags to an operation within an API.

Tags that are assigned to an operation can be used to group the operations of an API in the Developer Portal, by using the filter options for the API. These tags are visible to anyone who can view any of the Products to which the API belongs.

For more information, see [Editing an OpenAPI 2.0 API definition](#) and [Configuring an operation](#).

Note: Though you can tag the API itself in the API Setup section, by using this method the tag is not visible in the Developer Portal.

## Tags in the Developer Portal

An administrator can assign tags to items in the Developer Portal, which are then available through the API Products search function in the Developer Portal.

For more information, see [Editing tags for a specific item](#) and [Managing tags in the Developer Portal](#).

## Related concepts

- [Working with Products](#)

## Related tasks

- [Creating an API definition](#)

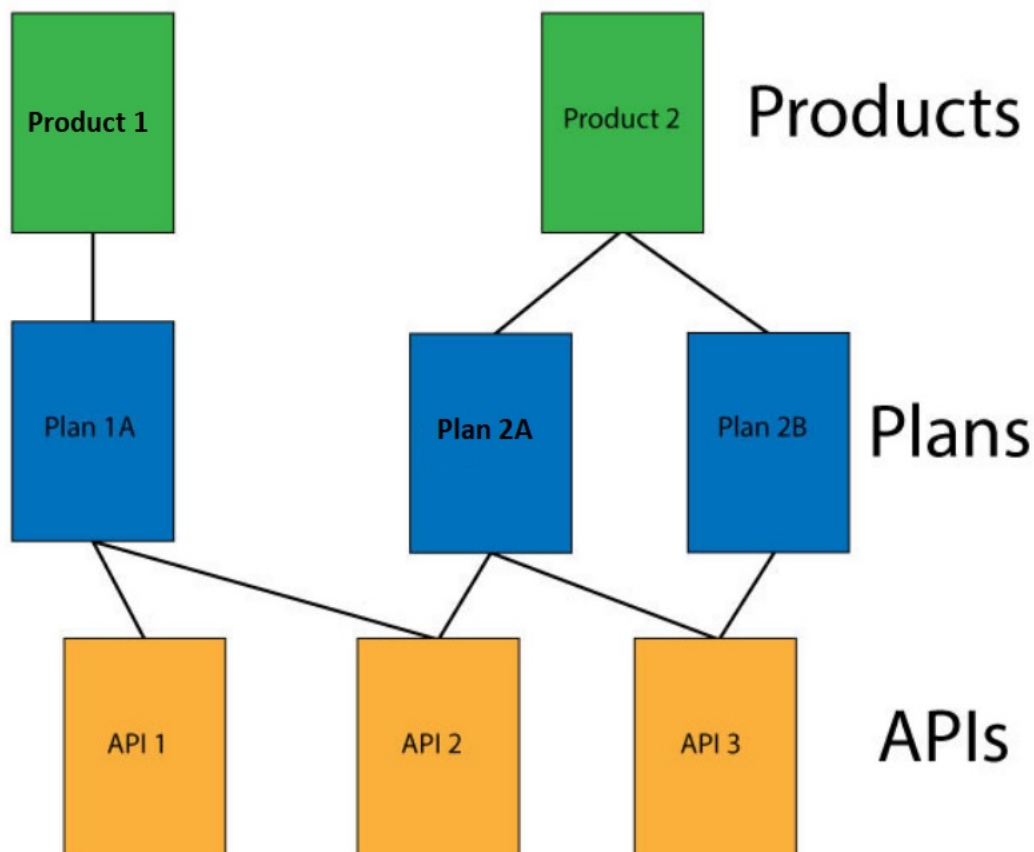
## Working with Products

In IBM® API Connect, Plans and APIs are grouped together in Products, with which you can manage the availability and visibility of APIs and Plans.

The following diagram demonstrates how Products, Plans, and APIs relate to one another.

Note: Plans belong to only one Product, but they can possess different APIs to other Plans within the same Product, and they can share APIs with Plans from any Product.

Figure 1. The hierarchy of Products, Plans, and APIs.



Products provide a method by which you can group APIs into a package that is intended for a particular use. Additionally, they contain Plans, which can be used to differentiate between different offerings. Plans can share APIs, but whether subscription approval is required depends upon the Plan itself. Additionally, you can enforce

rate limits through these Plans, or through operations within the APIs of a Plan that override the rate limit of the Plan. You can also assign different subscription costs to your Plans to differentiate the rates of API calls. For more information, see [Monetizing your Products](#).

To make an API available to an application developer, it must be included in a Plan. You can create Plans only within Products, and these Products are then published in a Catalog. A lifecycle manager can then control the availability and visibility of APIs and Plans through the API Manager. By using the API Manager, you can specify the Plans that an application developer is able to subscribe to through the Developer Portal. The application developer can only subscribe to one Plan from a specific Product. Multiple Plans within a single Product are useful in that they can fulfill similar purposes but with differing levels of performance and cost. For example, you can have a "Demo Plan", which makes a single API available free of charge, and a "Full Plan" which makes several APIs available with a monthly subscription cost.

As well as controlling which APIs an application developer can use, different Plans can be used to implement different rate limits. A rate limit can be implemented as a default rate that is shared across an entire Plan, or can be set for specific operations of an API within that Plan, exempting them from the shared Plan rate limit. Different Plans can have differing rate limits, both between operations and for the overall limit. This is useful for providing differing levels of service to customers. For example, a "Demo Plan" might enforce a rate limit of ten calls per minute, while a "Full Plan" might permit up to 1000 calls per second.

In addition, you can apply burst limits to your Plans, to prevent usage spikes that might damage infrastructure. Multiple burst limits can be set per Plan, at second and minute time intervals. You can also set multiple rate limits per Plan and per operation, at second, minute, hour, day, and week time intervals.

Note:

- Applying a rate limit at the Plan level creates a default rate limit that is shared across all the operations within the Plan. If you need to set specific rate limits for specific operations, you must set these within the operations themselves and these settings will override the setting at the Plan level.
- The test application that is used by the API Manager test tool is not subject to rate limits if you have enabled automatic subscriptions for the Catalog in which you are testing. For more information, see [Working with Catalogs](#)

API Connect also supports the implementation of multiple versions of Products. You can choose version numbers and use them to aid the development of your Products and Plans.

Note: The version for a Product is distinct from the version of any APIs that are contained in the associated Plans. Plans cannot themselves have their own version, they use the version of their parent Product.

- [Creating a draft Product](#)  
Create a draft Product to collect a set of APIs and Plans into one offering that you make available to your developers. A Plan includes rate limit settings that can be applied to the Plan as a whole, or specified for each operation in an API. Through Products and Plans, you have greater control over what APIs your developers have access to, and the subscription terms.
- [Downloading a draft Product](#)  
You can download the details of your draft Product so that you can store them for later recovery.
- [Importing a draft Product](#)  
You can create a Product by importing a Product definition in a .yaml file.
- [Searching for a draft Product](#)  
Quickly locate a draft API or Product by searching on the title and version.
- [Editing a draft Product](#)  
After initially creating a draft Product, you can continue to further configure the Product, or you can make configuration changes later.
- [Staging a draft Product](#)  
Stage a draft Product to a Catalog to create a specific version of that Product, before publishing. When a Product is in the staged state, it is not yet visible to, or subscribable by, any developers.. The syndication feature in IBM API Connect means that if Spaces are enabled for a Catalog, Products can be staged only to a Space within that Catalog.
- [Publishing a draft Product](#)  
Publish a draft Product to make the APIs in that Product visible on the Developer Portal for use by application developers. The syndication feature in IBM API Connect means that if Spaces are enabled for a Catalog, Products can be published only to a Space within that Catalog.
- [Creating a new version of your Product](#)  
You can have multiple versions of a Product in IBM API Connect. These versions can occupy any of the lifecycle stages, which facilitates development.
- [Specifying the gateway type for a Product](#)  
You can edit the Product configuration to specify a gateway type.
- [Deleting a draft Product](#)  
You can delete draft Products that are no longer required.
- [Organizing your Products into categories](#)  
You can organize your Products into categories. The Products that you categorize are displayed within the Developer Portal, in their defined categories.

## Related information

---

- [Working with Products in the API Manager](#)

---

## Creating a draft Product

Create a draft Product to collect a set of APIs and Plans into one offering that you make available to your developers. A Plan includes rate limit settings that can be applied to the Plan as a whole, or specified for each operation in an API. Through Products and Plans, you have greater control over what APIs your developers have access to, and the subscription terms.

## Before you begin

---

Define your APIs. For more information, see [Creating an API definition](#).

Tip: As well as using the method described in this task, you can also create a Product when you create an API. If you create an API by using the developer toolkit command line editor, a Product is automatically created for you. You can then change any of the Product settings by opening your new Product in the Products page of the API Designer.

## About this task

---


You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

When you first create your Product, it has *draft* status. When you have finished configuring your draft Product, you stage it to a Catalog so that the APIs in the Product can be made available to your application developers.

## Procedure

---

To create a Product, complete the following steps:

1. In the navigation pane, click  Develop, select the Products tab, then click Add Product.  
The Add Product page opens.
2. Select New Product, then click Next.
3. On the Info page, complete the following steps:
  - a. Enter the Title and Version of the Product and, optionally, a description. A Name is entered automatically.  
Note: The value in the Name field is a single string that is used to identify the Product in developer toolkit CLI commands. The Title is used for display. To view the CLI commands to manage draft Products, see [apic draft-products](#).
  - b. Click Next.
4. On the APIs page, select the APIs that you want to include in the Product. You can include only APIs whose gateway type matches the gateway type of the Product, or APIs for which the Enforced option is disabled, meaning that the API is not managed on an API Connect gateway. If you select an incompatible API whose gateway type does not match the gateway type of the Product, a warning message is displayed and you cannot save your changes until you clear the incompatible selection. For more information on gateway types, see [API Connect gateway types](#), [Specifying the gateway type for a Product](#), and [Specifying the gateway type for an API definition](#).
5. Click Next.
6. On the Plans page, complete the following steps:
  - a. Optional: Change the Title, Description, or Rate limit settings for the default Plan.
  - b. Optional: To add further Plans, click Add, then scroll down and modify the Plan settings as required.
  - c. Click Next.
7. To have the Product published to the Sandbox Catalog automatically after creation, select Publish product.  
Note: The publish option is not available in the API Designer UI.
8. If Publish product is selected, you can select which gateway service the Product is published to. By default, the Product is published to all relevant gateway services.
9. In the Visibility section, specify the users that you want the Product to be visible to. You can choose Public to make the Product visible to all users, Authenticated to make the Product available to users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that you want the Product to be visible to.  
If you select Custom, use the Type to add organizations field to search for the consumer organizations and consumer organization groups that can subscribe to the Plans in the Product. For information about how to create and manage consumer organizations and consumer organization groups, see [Administering Consumer organizations](#).  
  
Visibility can also be set at the Catalog or Space level. For more information, see [Configuring Product visibility in a Catalog](#).
10. In the Subscribability section, specify the users that can subscribe to the Plans in the Product. You can choose Authenticated to make the Plans in the Product subscribable by users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that can subscribe to the Plans in the Product.  
If you select Custom, use the Type to add organizations field to search for the consumer organizations and consumer organization groups that can subscribe to the Plans in the Product. If you selected custom visibility, only the consumer organizations and consumer organization groups selected there are available for adding to the custom subscribability list. For information about how to create and manage consumer organizations and consumer organization groups, see [Administering Consumer organizations](#).  
Note: If you select custom visibility or subscribability, the consumer organizations and consumer organization groups that you can choose from are those in the Sandbox Catalog for the management server and provider organization that you are connected to. If you want to specify consumer organizations or groups that are in another Catalog, or in a Space, publish the Product to that Catalog or Space, then configure the visibility or subscribability there; for details, see [Changing the availability of a Product](#).  
If you are using the API Manager user interface, the connection details are determined by the API Manager URL that you open, and the user ID with which you log in. If you are using the API Designer user interface, you provide the management server details and user ID in the login window that opens when you first launch API Designer; see [Logging into API Connect Designer](#).  
  
If you are working offline in API Designer, you must type in the exact name of the organization or group.  
  
For more information about consumer organizations in a Catalog, see [Administering Consumer organizations](#).
11. Click Next to create your Product.
  - a. To further configure your Product, click Edit Product.  
For details, see [Editing a draft Product](#)
  - b. If you do not want to configure your Product further at this time, click the initial Develop link in the breadcrumb trail to return to the Develop page; you can then move on immediately to another task.  
For details on how to configure your Product later, see [Editing a draft Product](#).

## Results

---

You have created a draft Product, and specified a set of APIs and Plans into one offering that you can now make available to your developers.

## What to do next

---

Stage your Product to a Catalog. For more information, see [Staging a draft Product](#).

## Related tasks

---

- [Creating a new version of your Product](#)

## Related reference

---

- [API and Product definition template examples](#)

## Related information

---

- [Publishing a Product](#)
- [Working with Products in the API Manager](#)
- [Configuring Product visibility in a Catalog](#)

---

## Downloading a draft Product

You can download the details of your draft Product so that you can store them for later recovery.

### Before you begin

---

To complete this task, you must have created a draft Product. For more information, see [Creating a draft Product](#).

### About this task

---

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.



**API Manager UI only:** To complete this task, you must be assigned a role that has the `Product-Drafts:Edit` permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see [Creating custom roles](#).

You can download the details of your Product as a .yaml file. This file can be used to re-create your Product, although it includes only references to APIs, not the API definitions themselves. This function does not provide a way to export an entire implementation. For information about creating a Product from a .yaml file, see [Importing a draft Product](#).

### Procedure

---

To download a .yaml file containing details of your Product, complete the following instructions:

1. In the navigation pane, click  Develop.
2. Select the Products tab.
3. Alongside the Product version that you want to work with, click the options icon  and then click Download.
4. Save the file to the required location.

### Results

---

You have downloaded a .yaml representation of your Product.

### Related tasks

---

- [Importing a draft Product](#)

### Related information

---

- [Creating a Product](#)

---

## Importing a draft Product

You can create a Product by importing a Product definition in a .yaml file.

### Before you begin

---

To complete this task, you must have a .yaml representation of a Product. This can be created as described in the [Downloading a draft Product](#) task.

### About this task

---

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.


You can import details of your Product as a .yaml file. This file can be used to re-create your Product, although it includes only references to APIs, not the API definitions themselves. This function does not provide a way to import an entire implementation.

You can import the file from your local file system.

### Procedure

---

To create a Product from a .yaml file, complete the following instructions:

1. In the navigation pane, click  Develop.
2. Select the Products tab.
3. Click Add > Product.
4. Select Existing Product, then click Next.
5. To import a file from your local file system, you can either drag and drop your files, or click Browse and select the file that you want to use. The wizard checks the validity of the YAML, and displays a message indicating successful validation.
6. Click Next.
7. To have the Product published to the Sandbox Catalog automatically after creation, select Publish product.
8. Click Next to create your Product.
  - a. To further configure your Product, click Edit Product. For details, see [Editing a draft Product](#)
  - b. If you do not want to configure your Product further at this time, click the initial Develop link in the breadcrumb trail to return to the Develop page; you can then move on immediately to another task. For details on how to configure your Product later, see [Editing a draft Product](#).

## Results

You have imported a Product from a .yaml file. If the same APIs and versions of APIs as referenced in the file are present, then they will be included in the Product.

Importing a Product in this way does not include the API definitions, only references to them.


## Related tasks

- [Downloading a draft Product](#)

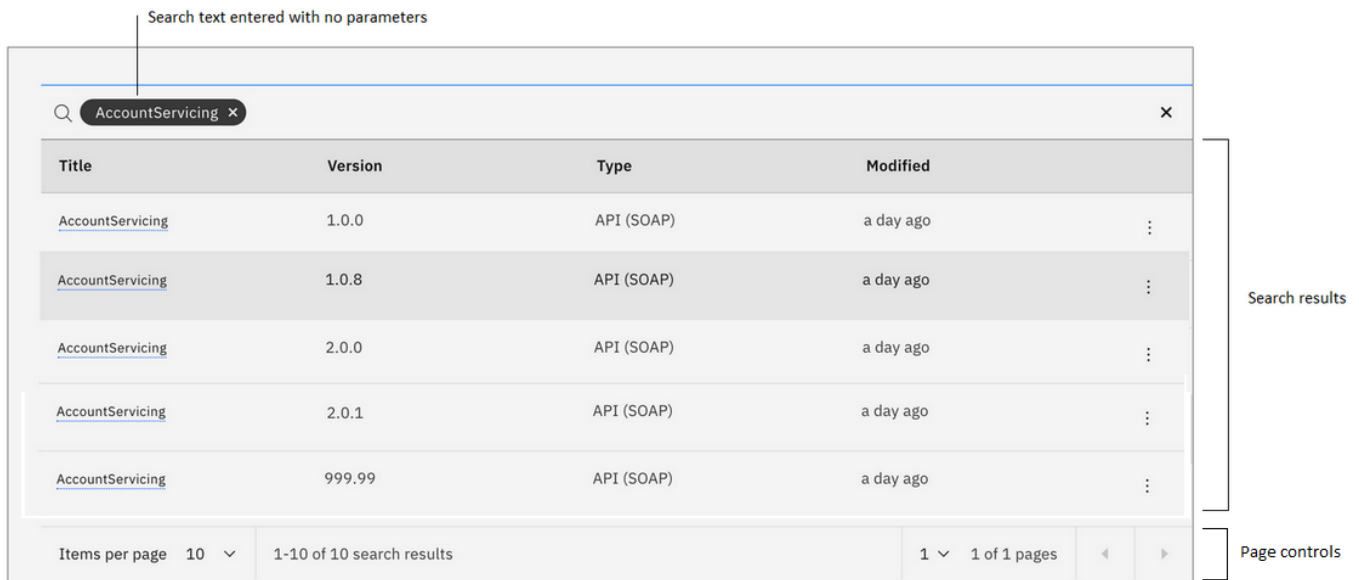
## Searching for a draft Product

Quickly locate a draft API or Product by searching on the title and version.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

On the Develop page, type your search string into the search field and click  (or press Enter) to run the search.

The search returns a list of APIs (or Products) that most closely resemble your keywords, rather than limiting the results to identical matches. The following image shows a search string that includes "AccountServicing". The results include all draft APIs and Products with a title that is similar to the search text, with duplicates sorted by creation date.



Search text entered with no parameters

Title	Version	Type	Modified
<a href="#">AccountServicing</a>	1.0.0	API (SOAP)	a day ago
<a href="#">AccountServicing</a>	1.0.8	API (SOAP)	a day ago
<a href="#">AccountServicing</a>	2.0.0	API (SOAP)	a day ago
<a href="#">AccountServicing</a>	2.0.1	API (SOAP)	a day ago
<a href="#">AccountServicing</a>	999.99	API (SOAP)	a day ago

Items per page 10 1-10 of 10 search results 1 1 of 1 pages

Search results

Page controls

Resources with a name, title, or version that are not similar to the search text are not included in the results. If you select a resource type to search for but do not specify any keywords, the results are sorted by title and then by creation date. For example, if you search for Products but do not include any keywords, then all Products appear in the results table.

### Search text

The search is case insensitive. Trailing spaces and accents (for example: é, è, â) are ignored. For best results, provide as much of the title and version as possible in your search text. Detailed search strings return more accurate results. Results include all titles that are similar to your search string, and not just identical matches.

### What's included in the search?

The scope of the search includes the API or Product Title, Name and Version fields. The search applies only to the current provider organization (displayed in the product banner's Organization field).

You can optionally restrict your search to Products only, or to a specific type of API. Click in the search field and then in the "Types" list that displays, select Product or an API type (REST or SOAP).



How are results displayed?

Search results display as a table, sorted by API or Product title and version. If the results overflow onto additional pages, navigate between pages using the standard page controls that follow the table. When you clear the search, APIs and Products are listed alphabetically by title and, within each title, by newest version first.

---

## Editing a draft Product

After initially creating a draft Product, you can continue to further configure the Product, or you can make configuration changes later.

---

### About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

During the initial creation of a Product, the Product creation wizard guides you to enter the minimum configuration settings, and then provides an Edit Product button. You have the option to specify additional configuration immediately, or to return to Product to edit it later. This topic describes how to specify additional configuration in either case.

---

### Procedure

To edit a draft Product, complete the following steps:

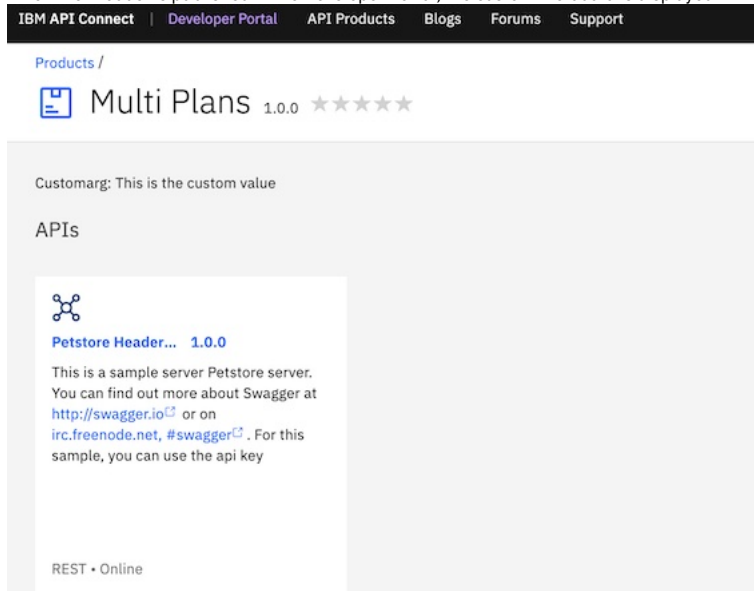
1. Access the Product Setup page in one of the following ways:
  - After completing the initial creation of the Product, click Edit Product.
  - In the navigation pane, click Develop, then select the Products tab.
  - Click the title of the Product that you want to edit.
2. On the Product Setup page, you can make the following configuration changes:
  - a. In the Info section, change the Title, Version, or Summary.
  - b. Use the Contact, Terms of Service, and License sections to enter the corresponding details as required.
  - c. Change the gateway type. For more information, see [API Connect gateway types](#) and [Specifying the gateway type for a Product](#).
  - d. Click Save to save your changes.
3. Optional: Specify Product metadata to be displayed with the Product in the Developer Portal:
  - a. Click the Source tab to see the OpenAPI source for the Product.
  - b. Add the custom attribute `x-name`:  
`value` into the `info` section. For example:

```
info:
 version: 1.0.0
 title: Multi Plans
 name: multi-plans
 x-customarg: This is the custom value
 ...
```

Where `customarg` is the name of the custom metadata, and `This is the custom value` is the information content.

c. Click Save.

When the Product is published in the Developer Portal, the custom metadata is displayed with the Product. For example:



4. Click Visibility, then make the following changes as required:
  - a. In the Visibility section, specify the users that you want the Product to be visible to. You can choose Public to make the Product visible to all users, Authenticated to make the Product available to users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that you want the Product to be visible to. If you select Custom, complete the following steps:
    - Select the Catalog for which you want to control visibility.
    - Use the Search organizations and groups field to search for the consumer organizations and consumer organization groups, in the selected Catalog, that you want the Product to be visible to. For information about how to create and manage consumer organizations and consumer organization

groups, see [Administering Consumer organizations](#).

- b. In the Subscribability section, specify the users that can subscribe to the Plans in the Product. You can choose Authenticated to make the Plans in the Product subscribable by users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that can subscribe to the Plans in the Product.

If you select Custom, complete the following steps:

- Select the Catalog for which you want to control subscribability.
- Use the Search organizations and groups field to search for the consumer organizations and consumer organization groups, in the selected Catalog, that you want the Product to be subscribable to. If you selected custom visibility, only the consumer organizations and consumer organization groups selected there are available for adding to the custom subscribability list. For information about how to create and manage consumer organizations and consumer organization groups, see [Administering Consumer organizations](#).

Note:

- If you select custom visibility or subscribability, only 10 search results are displayed in the selection list for the consumer organizations and consumer organization groups. If necessary, enter a more specific search string to refine the search.
- Although the title of each organization or group is displayed, followed by the name in parentheses, in the table of selected organizations and groups only the name is displayed

- a. Click Save to save your changes.

Note: You can change the visibility and subscribability settings when you publish the Product; see [Publishing a draft Product](#). You can also change the settings in the Catalog to which the Product is published; see [Changing the availability of a Product](#).

5. To specify the APIs that you want to include in the Product, click APIs, then complete the following steps:.

- a. Click Edit.

All draft APIs are listed.

- b. Select the APIs that you want to include. You can include only APIs whose gateway type matches the gateway type of the Product, or APIs for which the Enforced option is disabled, meaning that the API is not managed on an API Connect gateway. If you select an incompatible API whose gateway type does not match the gateway type of the Product, a warning message is displayed and you cannot save your changes until you clear the incompatible selection. For more information on gateway types, see [API Connect gateway types](#), [Specifying the gateway type for a Product](#), and [Specifying the gateway type for an API definition](#).

- c. Click Save when done.

The selected APIs are listed.

Note: To make an API available to application developers, you must include it in a Plan.


6. Optional: Add billing integration to the Product. Click Plans, and in the Billing integration section, select the billing integration resource that you want to apply to the Product.

In the Plans section, you can then either edit the Default Plan to add pricing information, or create a new Plan with pricing information. For more information about billing integration, see [Monetizing your Products](#).

7. Optional: Add one or more Plans to the Product, or modify an existing Plan. Note that a default plan is automatically created for you, with a rate limit of 100 API calls per hour.

Note: If you enable event endpoints in Cloud Pak for Integration, rate limits do not apply to AsyncAPIs associated with a plan.

- a. Click Plans.

- b. To add a new Plan, click Add. To modify an existing Plan, click the options icon  alongside the required Plan, then click Edit.

- c. Specify the Title of the Plan and, optionally, a description. A Name is entered automatically.


Note: The value in the Name field is a single string that is used to identify the Plan in developer toolkit CLI commands. The Title is used for display.

- d. Specify whether your Plan requires subscription approval. If you want subscriptions by developers to require approval, select the Approval check box; otherwise, ensure the check box is cleared.

- e. In the Plan pricing section, you can add pricing information by completing the following steps:


- i. Change the toggle to On for Plan pricing. The Plan pricing definition section is displayed.
- ii. If you want to include any free trial days in your Plan, select Include free trial days, and then enter the number of trial days that a subscriber can use the Plan without charge, after which their billing cycle begins.
- iii. Select the Currency for the billing process use.
- iv. Enter the Price per month to bill the subscriber for. If the selected currency supports fractional units, enter the price including any fractional units, such as cents.

For more information about Product billing integration, see [Defining a Product with billing integration](#).

- f. In the Plan Rate Limits section, you can modify a rate limit, and click Add to add further rate limits. You can set multiple rate limits, at second, minute, hour, day, and week time intervals. To remove a rate limit, click the corresponding delete icon .

Note: For information about rate limits and burst limits in API Connect, see [Understanding rate limits for APIs and Plans](#).

- g. In the Plan Burst Limits section, you can modify a burst limit, and click Add to add further burst limits. You use burst limits to prevent usage spikes that might

damage infrastructure. You can set multiple burst limits, at second and minute intervals. To remove a burst limit, click the corresponding delete icon . Rate and burst limits work together to manage network traffic for the APIs covered under a Plan. A Plan can have multiple rate and burst limits, but it is recommended that each time interval be assigned only one set of limits. Adjust the rate and burst limits to allow for maximum traffic without overloading your network. The **rate limit** sets the maximum amount of sustainable, ongoing traffic for accessing the APIs on your network within a time interval (for example, one day). The **burst limit** sets the maximum short-term traffic volume for your network within a time interval (per second or minute).

The **burst limit** allows for short bursts of increased traffic. When the **burst limit** is exceeded, all subsequent API calls are rejected until the start of the next **burst limit interval**. The **burst limit counter** is reset to zero at the start of the next interval, which allows API calls to be accepted again. These API calls count toward the **rate limit counter**, but the resetting of the **burst limit counter** does not affect the **rate limit counter**.

The **rate limit** is the number of API calls allowed in a time interval, for example, 1000 calls per day. When the **rate limit** is exceeded, and Hard limit is enabled, all subsequent API calls will be rejected. The **rate limit counter** is reset to zero at the start of the next **rate limit interval**, which allows API calls to be accepted again. If Hard limit is disabled, all subsequent API calls are still accepted, and a message is logged stating that the **rate limit** has been exceeded. This is referred to as a "soft limit."

Hard limit affects only the **rate limit**, as illustrated by the following scenarios:

Scenario A

Table 1. Hard limit enabled

Hard limit	Burst limit	Rate limit
ON	100 calls/minute	1000 calls/day

- If a user calls an API 100 times in one minute, the **burst limit** is reached. The 101st call (and any subsequent calls) within the same minute will be rejected. Once the minute is up, the **burst limit counter** is reset. All API calls are tallied toward the **rate limit** of 1000 calls per day. The resetting of the **burst limit counter** does not affect the **rate limit counter**.

- If a user calls the API 99 times per minute, they will not hit the **burst limit**. They will eventually hit the 1000 calls per day rate limit, and the 1001st call will be rejected until the end of the same one day **rate limit interval**. During the period of time when API calls are rejected due to the daily **rate limit** being exceeded, the **burst limit** will not be activated since the calls are already rejected.
- Both **burst limits** and **rate limits** are applied per consumer.

Scenario B

Table 2. Hard limit disabled

Hard limit	Burst limit	Rate limit
OFF	100 calls/minute	1000 calls/day

- As in Scenario A, if a user calls the API 100 times in one minute, the 101st call within the same minute will be rejected, until that minute is up and the counter resets. These calls count toward the 1000 calls per day **rate limit** as well.
- If a user calls the API 99 times per minute, they will not hit the **burst limit**. They will eventually hit the 1000 calls per day **rate limit**, and the 1001st call will be accepted (since there is no hard limit). A message will be logged for each subsequent call until the time interval (one day) is up and the counter resets. During the remainder of the day, the **burst limit** will still be enforced, and calls will be rejected once the number of calls exceeds the 100 calls per minute within any given minute.
- Both burst limits and rate limits are applied per consumer.

Note: When Hard limit is unchecked, the **rate limit** is considered a "soft limit." With a soft limit, calls are not rejected after the **rate limit** is reached. Instead, a message is recorded in the log file. With a soft limit, the **burst limit** still rejects API calls after it is exceeded.

- To include in the Plan all the APIs that were included in the Product in step 5, select Same as product in the Plan APIs section.
- In the GraphQL Rate Limits section you can configure rate limits that will be applied to any GraphQL proxy APIs that are added to the Product. The following rate limits are available:

graphql-field-cost

Applies a limit to the total calculated field cost of GraphQL query calls made to the GraphQL proxy APIs in this Plan.

graphql-design-request

Applies a limit to the total number of calls of any of the following types made to the GraphQL proxy APIs in this Plan:

- Requests sent to the **graphql/cost** endpoint for retrieving the cost of a GraphQL query. For more information, see [Enabling the cost endpoint for a GraphQL API](#).
- Standard introspection requests sent to the **/graphql** endpoint. For more information, see [Supporting introspection for a GraphQL API](#).
- HTML web browser requests for a GraphQL editor sent to the **/graphql** endpoint. For more information, see [Enabling the GraphQL editor for a GraphQL API](#).




graphql-input-type-cost

Applies a limit to the total calculated input type cost of GraphQL query calls made to the GraphQL proxy APIs in this Plan.

graphql-type-cost

Applies a limit to the total calculated type cost of GraphQL query calls made to the GraphQL proxy APIs in this Plan.

The calculated costs depend on weighting factors applied to the types and fields in the GraphQL schema. For details on the GraphQL proxy API, and configuring type and field weights, see [Creating a GraphQL proxy API](#) and [Using the GraphQL schema editor](#).

- To define one or more assembly burst limits or assembly count limits, click Add in the Assembly Burst Limits or Assembly Count Limits section as appropriate. For a burst limit you specify the maximum number of calls allowed in a specified time period, for a count limit you specify a maximum call limit. The name that you supply is for use in a Rate Limit policy in an API assembly, enabling the policy to define the burst or count limit that is to be applied; for more information, see [Rate Limit](#).
- To select the APIs that you want to include in the Plan, select Customize the plan API list in the Plan APIs section, then click Edit. The Edit plan APIs window opens. The APIs that are available for selection are those that were included in the Product in step 5. Use the check boxes to specify the required APIs, then click Save when done.
- To select specific API operations to include in the Plan, click the options icon  alongside the required API, then click Edit operation list. The Select operations you want to include window opens. Use the check boxes to specify the required operations, then click Save when done.
- To remove an API from the Plan, click the options icon  alongside the required API in the Plan APIs section, then click Remove.
- For any GraphQL APIs in the Plan, if you want to prevent subscribers to this Plan from using certain types or fields, complete the following steps:
  - Click the options icon  alongside the required GraphQL API in the Plan APIs section, then click Edit show/hide settings.
  - To prevent use of a GraphQL type, clear the check box selection alongside that type.
  - To prevent use of a field in a type, expand the type then clear the check box selection alongside that field.
  - When done, click Next. A summary window opens. If the operation is permitted, any related elements that will also be hidden are listed; for example, if you hide a type, and that type is referenced as the data type of a field on another type, that field is also hidden. If the operation is not permitted, an explanation is displayed.
  - If the operation is permitted, click Done to apply your changes. If the operation is not permitted, either click Cancel to close the window or click Back and amend your settings.

You can use the Edit show/hide settings option to change existing settings; you can hide additional types or fields, and you can show types or fields that were previously hidden.

- To define rate limits for specific API operations, select Override plan rate limits for individual operation in the Plan APIs section; for further details, see [Defining rate limits for an API operation](#)
- Click Save to save your changes.

8. Optional: Specify Plan metadata to be displayed with the Plan in the Developer Portal:

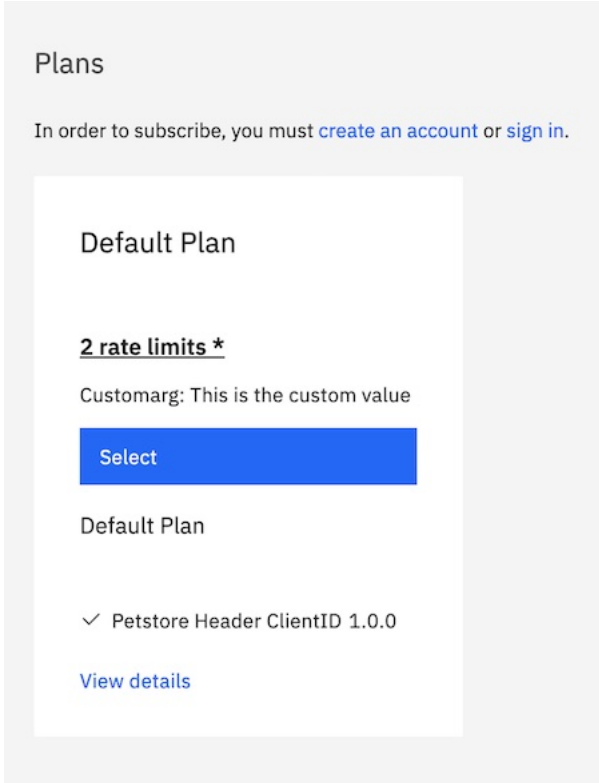
- Click the Source tab to see the OpenAPI source for the Plan.
- Add the custom attribute **x-name**: **value** into the appropriate **plan** section. For example:

```
...
plans:
 default-plan:
 title: Default Plan
 description: Default Plan
 x-customarg: This is the custom value
 rate-limits:
 default:
 value: 100/1hour
 hard-limit: false
 minute:
```

```
value: 20/1minute
hard-limit: false
apis:
 petstore-header-clientid1.0.0: {}
...
```

Where *customarg* is the name of the custom metadata, and *This is the custom value* is the information content.  
c. Click Save.

When the Plan is published in the Developer Portal, the custom metadata is displayed with the Plan. For example:



Note: After adding custom metadata to a Plan, if you need to edit the Plan details further, you must do this in the Source view, or the custom metadata will be deleted.

9. Optional: Click Categories, then define any categories that you want to organize your Product into. Click Save to save any updates.

By organizing your Products into categories, you can provide a hierarchical display for your Products in the Developer Portal. For more information, see [Organizing your Products into categories](#).

## What to do next

---

Stage your Product to a Catalog. For more information, see [Staging a draft Product](#).

- [Defining rate limits for an API operation](#)  
For any Plan in a draft Product, you can apply one or more rate limits to a specific operation on any of the APIs in that Plan.

## Related tasks

---

- [Creating a new version of your Product](#)

## Related reference

---

- [API and Product definition template examples](#)

## Related information

---

- [Publishing a Product](#)
- [Working with Products in the API Manager](#)

## Defining rate limits for an API operation

For any Plan in a draft Product, you can apply one or more rate limits to a specific operation on any of the APIs in that Plan.

## About this task

---



You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

During the initial creation of a Product, the Product creation wizard guides you to enter the minimum configuration settings, and then provides an Edit Product button. You have the option to specify additional configuration immediately, or to return to Product to edit it later. This topic describes how to specify additional configuration in either case.

Note: For information about rate limits and burst limits in API Connect, see [Understanding rate limits for APIs and Plans](#).

## Procedure

---

1. Access the Product Setup page in one of the following ways:
  - After completing the initial creation of the Product, click Edit Product.
  - In the navigation pane, click Develop, then select the Products tab.
  - Click the title of the Product that you want to edit.
2. Click Plans.
3. Click the options icon  alongside the required Plan, then click Edit.
4. Scroll down to the Plan APIs section, then select Override plan rate limits for individual operation. The Override Rate Limits section is displayed.
5. Click Add. The Override plan rate limits window opens.
6. Select the API, and the specific operation in that API, to which you want to apply rate limits. By default, calls to the operation are unlimited.
7. To apply one or more rate limits to the operation, select Rate Limits.
8. To define a rate limit, click Add.
9. Supply a name for the rate limit, and define the maximum number of calls allowed in a specified time period; for example, 100 calls per 1 minute. If you select HARD LIMIT and the limit is exceeded, subsequent calls are rejected, otherwise calls are still accepted and a message is logged.
10. Click Save when done. An entry for the operation is listed in the Override Rate Limits section.
11. To modify the rate limiting definition for an operation, click the options icon  alongside the required operation, then click Edit Operation; to delete it, click Remove.
12. Click Save to save your changes.

---

## Staging a draft Product

Stage a draft Product to a Catalog to create a specific version of that Product, before publishing. When a Product is in the staged state, it is not yet visible to, or subscribable by, any developers.. The syndication feature in IBM® API Connect means that if Spaces are enabled for a Catalog, Products can be staged only to a Space within that Catalog.

## Before you begin

---

Ensure that you have a Catalog to stage to in the API Manager or API Designer user interfaces (UI). For more information, see [Creating and configuring Catalogs](#).

Note: All references in this topic to a Catalog can also be applied to Spaces in a Catalog, unless specified otherwise. For more information about Spaces, see [Using syndication in IBM API Connect®](#).

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Product > Stage permission for the target Catalog or Space. For information on configuring Product management permissions for a Catalog or Space, see [Creating and configuring Catalogs](#) or [Managing user access in a Space](#).

## About this task

---

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI. Staging is not available when working offline in API Designer.

A Catalog is a staging target, and behaves as a logical partition of the DataPower® Gateway, and the Developer Portal.

You stage a Product so that the appropriate approvals, internally within the organization, can be given for it to then be published. For information on configuring Product lifecycle approvals for a Catalog, see [Creating and configuring Catalogs](#). For information on approving requests, see [Approving Product lifecycle and subscription requests](#). For information on publishing a Product, see [Publishing a Product](#).

You cannot re-stage an existing Product version to a production Catalog. If you update an API, you must create a new version of that API. Then, create a new version of the Product so that it includes the newer version of the API. For more information on creating new versions of APIs and Products, see [Creating a new version of an API definition](#) and [Creating a new version of your Product](#). If Spaces are enabled in a production Catalog, you cannot re-stage the same API or Product version to any Space in the Catalog.

You can, however, re-stage a Product version to the Sandbox Catalog, or other non-production Catalog, for ease of iterative testing. Note that when you re-stage a Product version to a non-production Catalog, any application subscriptions are deleted; this is done to facilitate scripted deployment in a continuous integration environment, where the re-creation of application subscriptions will be controlled by the scripts. If Spaces are enabled in a non-production Catalog and you re-stage a Product version to a different Space to the one in which it was previously staged, it is removed from the previous Space and staged to the newly specified Space. If billing integration is enabled in a Product, you cannot re-stage the same Product version into a non-production, or production, Catalog. For more information about billing integration, see [Monetizing your Products](#).

Validation of the API definition file occurs during the staging or publishing process. The following validation occurs:

- Validation against the OpenAPI schema by using the API Dev Tools Swagger Parser (<https://www.npmjs.com/package/@apidevtools/swagger-parser>).
- Validation against IBM extension properties.
- Semantic validation, which includes the following types of validation:
  - Ensuring that if an OpenAPI is enforced by an API Connect Gateway, then the scheme must be HTTPS, or the parameter name for an API key security scheme in the header must be either **X-IBM-Client-Id** or **X-IBM-Client-Secret**.
  - Ensuring that if the OpenAPI is not enforced by an API Connect Gateway, then a "host" must be provided.

- De-referencing of local references in the definition file (that is, values of `$ref` properties), and ensuring these are valid JSON Pointers within the file.



Note: If you are using Event Endpoint Management in Cloud Pak for Integration, similar validation is carried out for AsyncAPI documents. For more information, see the section about API document and enforcement in the [Cloud Pak for Integration documentation](#).

Note: If the OpenAPI file that defines your API uses a `$ref` field to reference a fragment of OpenAPI code that is defined in a separate file, the `$ref` field is replaced with the contents of the target file before the product that contains the API is staged or published (the `$ref` field is supported only if you are using the API Connect for IBM Cloud developer toolkit). For more information, see [Using \\$ref to reuse code fragments in your OpenAPI files](#).

## Procedure

---

To stage a Product, complete the following steps:

1. **Optional:** If you have accounts on multiple provider organizations, you can select a new provider organization for staging and publishing from the Organization menu.
2. In the navigation pane, click  Develop.
3. Select the Products tab.
4. You can stage a Product either from the Develop listing page, or from within the Product itself.
  - a. To stage a Product from the Develop listing page, click the options menu icon  alongside the required Product version, and then select Stage.
  - b. To stage a Product from within the Product itself, complete the following steps:
    - i. Click the Product version that you want to work with.
    - ii. Click the options menu icon:




- iii. Select Stage.
5. Select the Catalog to which you want to stage the Product.
  6. If Spaces have been enabled in the selected Catalog, select the Space that you require.

Note: The Catalogs that you can select from are those that are defined for the management server and provider organization that you are connected to. If you are using the API Manager user interface, the connection details are determined by the API Manager URL that you open, and the user ID with which you log in. If you are using the API Designer user interface, you provide the management server details and user ID in the login window that opens when you first launch API Designer; see [Logging into API Connect Designer](#).

For details of how to create a Catalog in a provider organization, see [Creating and configuring Catalogs](#).
  7. If, when the staged Product is subsequently published, you want it to be published only to selected gateway services, select Publish to specific gateway services, then select the required gateway services. Only the gateway services whose type matches the gateway type setting for the Product are listed. For information on gateway types, see [API Connect gateway types](#).
  8. Click Stage.

## Results

---

Your Product is staged to a Catalog. You can view the state of the Product in the Catalog in API Manager. If you staged the product from API Designer, ensure you are logged into API Manager with the same user name and password that you used for API Designer. Click  Manage in the API Manager UI, then select the required Catalog. The Product is shown with a state of Staged.

For information about the lifecycle of a product, see [The Product lifecycle](#).

If approval is required to stage Products in the Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is staged when the request is approved. If approval is not required, the Product is staged immediately.

For information on configuring Product lifecycle approvals for a Catalog, see [Creating and configuring Catalogs](#). For information on approving requests, see [Approving Product lifecycle and subscription requests](#).

If the Product contains no Plans, a Plan called Default Plan is added automatically to the Product in the Catalog.

## What to do next

---

- Publish your Product for application developers to access it in the Developer Portal. For more information, see [Publishing a Product](#).

## Related information

---

- [Managing your Products](#)
- [Removing a Product from a Catalog](#)
- [Working with Products in the API Manager](#)

## Publishing a draft Product

---

Publish a draft Product to make the APIs in that Product visible on the Developer Portal for use by application developers. The syndication feature in IBM® API Connect means that if Spaces are enabled for a Catalog, Products can be published only to a Space within that Catalog.

## Before you begin

---

Ensure that you have a Catalog to stage to in the API Manager or API Designer user interfaces (UI). For more information, see [Creating and configuring Catalogs](#).

Note: All references in this topic to a Catalog can also be applied to Spaces in a Catalog, unless specified otherwise. For more information about Spaces, see [Using syndication in IBM API Connect®](#).

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Product Management permission for the target Catalog or Space. For information on configuring Product management permissions for a Catalog or Space, see [Creating and configuring Catalogs](#) or [Managing user access in a Space](#).

## About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI. Publishing is not available when working offline in API Designer.

You cannot re-publish a Product version to a production Catalog, you must create a new version of the Product for publication; see [Creating a new version of your Product](#). If Spaces are enabled in a production Catalog, you cannot re-publish the same Product version to any Space in the Catalog, neither the same Space nor a different Space.

You can, however, re-publish a Product version to the Sandbox Catalog, or other non-production Catalog, for ease of iterative testing. When you re-publish a Product version to a non-production Catalog, you can choose whether or not to preserve any application subscriptions; deleting subscriptions can facilitate scripted deployment in a continuous integration environment, where the re-creation of application subscriptions will be controlled by the scripts.

If Spaces are enabled in a non-production Catalog and you re-publish a Product version to a different Space to the one in which it was previously published, it is removed from the previous Space and published to the newly specified Space.

Validation of the API definition file occurs during the staging or publishing process. The following validation occurs:



- Validation against the OpenAPI schema by using the API Dev Tools Swagger Parser (<https://www.npmjs.com/package/@apidevtools/swagger-parser>).
- Validation against IBM extension properties.
- Semantic validation, which includes the following types of validation:
  - Ensuring that if an OpenAPI is enforced by an API Connect Gateway, then the scheme must be HTTPS, or the parameter name for an API key security scheme in the header must be either `X-IBM-Client-Id` or `X-IBM-Client-Secret`.
  - Ensuring that if the OpenAPI is not enforced by an API Connect Gateway, then a "host" must be provided.
  - De-referencing of local references in the definition file (that is, values of `$ref` properties), and ensuring these are valid JSON Pointers within the file.

Note: If you are using Event Endpoint Management in Cloud Pak for Integration, similar validation is carried out for AsyncAPI documents. For more information, see the section about API document and enforcement in the [Cloud Pak for Integration documentation](#).


Note: If the OpenAPI file that defines your API uses a `$ref` field to reference a fragment of OpenAPI code that is defined in a separate file, the `$ref` field is replaced with the contents of the target file before the product that contains the API is staged or published (the `$ref` field is supported only if you are using the API Connect for IBM Cloud developer toolkit). For more information, see [Using \\$ref to reuse code fragments in your OpenAPI files](#).

## Procedure

To publish a Product, complete the following steps:

1. **Optional:** If you have accounts on multiple provider organizations, you can select a new provider organization for staging and publishing from the Organization menu.
2. In the navigation pane, click  Develop.
3. Select the Products tab.
4. You can publish a Product either from the Develop listing page, or from within the Product itself.
  - a. To publish a Product from the Develop listing page, click the options menu icon  alongside the required Product version, and then select Publish. If you are re-publishing a Product to non-production Catalog, and the Product has application subscriptions, the Publish option deletes them. To preserve application subscriptions, click Publish (Preserve Subscriptions) instead.

Note: If there are any existing subscriptions to the Product, the Publish (Preserve Subscriptions) operation will fail if you remove any consumer organizations or consumer organization groups from the Product visibility or subscribability settings; you can, however, add further organizations or groups. For information on managing visibility and subscribability settings, see [Editing a draft Product](#).

For information on consumer organization groups, see [Working with consumer organization groups](#).
  - b. To publish a Product from within the Product itself, complete the following steps:
    - i. Click the Product version that you want to work with.
    - ii. Click the options menu icon:  

    - iii. Select Publish.
5. Select the Catalog to which you want to publish the Product. You must have a compatible gateway type associated with the catalog to proceed, as described in [Creating and configuring Catalogs](#).
6. If Spaces have been enabled in the selected Catalog, select the Space that you require.

Note: The Catalogs that you can select from are those that are defined for the management server and provider organization that you are connected to. If you are using the API Manager user interface, the connection details are determined by the API Manager URL that you open, and the user ID with which you log in. If you are using the API Designer user interface, you provide the management server details and user ID in the login window that opens when you first launch API Designer; see [Logging into API Connect Designer](#).

For details of how to create a Catalog in a provider organization, see [Creating and configuring Catalogs](#).
7. If you want to publish the Product only to selected gateway services, select Publish to specific gateway services, then select the required gateway services. Only the gateway services whose type matches the gateway type setting for the Product are listed. For information on gateway types, see [API Connect gateway types](#).
8. The Visibility and Subscribability sections display the visibility and subscribability settings that have been configured for the draft Product, but you can modify these before publishing if required; proceed as follows:
  - a. In the Visibility section, specify the users that you want the Product to be visible to. You can choose Public to make the Product visible to all users, Authenticated to make the Product available to users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that you want the Product to be visible to. If you select Custom, use the Type to add organizations field to search for the consumer organizations and consumer organization groups, in the selected Catalog, that you want the Product to be visible to. For information about how to create and manage consumer organizations and consumer organization groups, see [Administering Consumer organizations](#).

- b. In the Subscribability section, specify the users that can subscribe to the Plans in the Product. You can choose Authenticated to make the Plans in the Product subscribable by users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that can subscribe to the Plans in the Product.

If you select Custom, use the Search organizations and groups field to search for the consumer organizations and consumer organization groups, in the selected Catalog, that you want the Product to be subscribable to. If you selected custom visibility, only the consumer organizations and consumer organization groups selected there are available for adding to the custom subscribability list. For information about how to create and manage consumer organizations and consumer organization groups, see [Administering Consumer organizations](#).

Note:


- If you select custom visibility or subscribability, only 10 search results are displayed in the selection list for the consumer organizations and consumer organization groups. If necessary, enter a more specific search string to refine the search.
- If custom visibility or subscribability has already been configured in the draft Product, the titles that are displayed in the table of selected organizations and groups are derived by taking the names that are defined in the draft Product and finding the corresponding titles in the Catalog that has been selected as the publish target.

9. Click Publish.

## Results

---

Your Product is published to a Catalog. The state of the Product is shown as Published in the header of the Product itself, and you can click Published to see which Catalogs or Spaces the Product is published to.

You can also view the state of the Product in the Catalog in API Manager. If you published the product from API Designer, ensure you are logged into API Manager with the same user name and password that you used for API Designer. Click  Manage in the API Manager UI, then select the required Catalog. The Product is shown with a state of Published.

For information about the lifecycle of a product, see [The Product lifecycle](#).

If approval is required to publish Products in the Catalog, a publish approval request is sent, and the Product moves to the Pending state; the Product is published in the Catalog when the request is approved. If approval is not required, the Product is published immediately.

Note: If approval is required to stage Products to the Catalog, a stage approval request is sent. When the request is approved, the Product moves to the staged state and must be separately published in the Catalog. For information on publishing a staged Product in a Catalog, see [Publishing a Product](#).

For information on configuring Product lifecycle approvals for a Catalog, see [Creating and configuring Catalogs](#). For information on approving requests, see [Approving Product lifecycle and subscription requests](#).

If the Product contains no Plans, a Plan called Default Plan is added automatically to the Product in the Catalog.

## Related information

---

- [Managing your Products](#)
- [Removing a Product from a Catalog](#)
- [Working with Products in the API Manager](#)

## Creating a new version of your Product

You can have multiple versions of a Product in IBM® API Connect. These versions can occupy any of the lifecycle stages, which facilitates development.

## Before you begin

---

Create your Product and assign it a version. For more information, see [Creating a draft Product](#).

## About this task

---



You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can create multiple versions of a Product, and these versions can be named according to your own preferences, and function as distinct Products when staged.

## Procedure

---

To create a new version of a Product, complete the following steps:

1. In the navigation pane, click  Develop.
2. Select the Products tab.
3. Alongside the Product version that you want to work with, click the options icon , then click Save as a new version. The Save Product as a new version window opens.
4. Enter your new version number, then click Save.

## Results

---

Your Product is saved as a new version.

## Related information

---

- [Managing your Products](#)



- [Working with Products in the API Manager](#)

## Specifying the gateway type for a Product

You can edit the Product configuration to specify a gateway type.

### Before you begin

IBM® API Connect supports two gateway types: DataPower® Gateway (v5 compatible) and DataPower API Gateway.

DataPower Gateway (v5 compatible) has been available with IBM API Connect for a number of years. The DataPower API Gateway is a new gateway that has been designed with APIs in mind, and with the same security focus as DataPower Gateway (v5 compatible).

In addition, Event Endpoint Management in Cloud Pak for Integration provides the Event Gateway Service for handling incoming requests to consume from a stream of events.

For more information on how to choose which gateway type to use, see [API Connect gateway types](#).

### About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can specify the gateway type for a Product.

Products can use only one type of gateway, and the APIs in the Product must use the same type of gateway. See [Specifying the gateway type for an API definition](#).

When you create a new Product, or import a Product definition that has no gateway type configured, the gateway type defaults according to the gateway types of the gateway services that are enabled in your Sandbox Catalog, as indicated in the following table.

Table 1. Default gateway type for a new Product


Gateway Service enablement in Sandbox Catalog	Default gateway type for new Product
No gateway services enabled	DataPower API Gateway
At least one gateway service of each type enabled	DataPower API Gateway
One or more gateway services only of type DataPower API Gateway enabled	DataPower API Gateway
One or more gateway services only of type DataPower Gateway (v5 compatible) enabled	DataPower Gateway (v5 compatible)

If you import a Product definition that already has a gateway type configured, that gateway type is retained.

For details of how to enable gateway services in a Catalog, see [Creating and configuring Catalogs](#).

### Procedure

To specify the gateway type for a Product, complete the following steps:

1. Ensure that you know the type of gateway that your APIs are configured to use.
2. In the navigation pane, click  Develop, then select the Products tab.
3. Click the title of the Product you want to update.
4. On the Product Setup page, go to the Gateway Type section. Select the gateway type for your Product:
  - DataPower Gateway (v5 compatible)
  - DataPower API Gateway
  - Event Gateway

Note:

You can only select the gateway type that satisfies the selected enforced APIs in your Product. To change the APIs included in your product, see [Editing a draft Product](#).

5. Click Save to retain your changes.

## Deleting a draft Product

You can delete draft Products that are no longer required.

### About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.



**API Manager UI only:** To complete this task, you must be assigned a role that has the **Product-Drafts:Edit** permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see [Creating custom roles](#).

Note:

- Deleting a draft Product does not remove it from any Catalogs to which it has been published. For details on how to remove a Product from a Catalog, see [Removing a Product from a Catalog](#).
- You cannot directly delete a Product that was generated automatically from an API with the Activate API option; these Products have the title *api\_title* auto product. To delete such Products, delete the API from which it was generated; the Product is deleted together with the API. For more information, see [Deleting an API definition](#).

## Procedure

---

1. In the navigation pane, click  Develop.
2. Select the Products tab.
3. Alongside the Product version that you want to delete, click the options icon  and then click Delete.

---

## Organizing your Products into categories

You can organize your Products into categories. The Products that you categorize are displayed within the Developer Portal, in their defined categories.

### Before you begin

---

Your role must have the necessary permissions to stage and publish Products.

### About this task

---


You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

By organizing your Products into categories, you can provide a hierarchical display for your Products in the Developer Portal.

Note: Organizing Products into a hierarchical view in the API Designer or API Manager UI is different to tagging in the Developer Portal.

## Procedure

---

1. In the navigation pane, click  Develop, then select the Products tab..
2. Click the title of the Product that you want to work with.
3. Click Categories.
4. In the Categories text box, enter the hierarchical taxonomy path that you want your Product to follow, in the following format:

```
top_level_element / next_level_element / x_level_element
```

For example:

```
Animals / Fluffy / Cat
```

5. After you have completed entering the category specifications for the Product, click Save.

### What to do next

---

- Publish the Product; see [Publishing a draft Product](#).
- After publishing the Product, you can, in the Developer Portal, display Products in the categories that you have defined. For more information, see [Displaying APIs and Products in categories](#).

---

## Creating and validating API and Product definitions by using the CLI

The developer toolkit of IBM® API Connect provides a command line interface that you can use to create and publish API and Product definitions, and also to validate YAML or JSON definitions.

The following topics explain how to use the command line interface to create an OpenAPI definition file, create a Product definition file, and validate a YAML or JSON definition.

- [Creating an OpenAPI definition file](#)  
APIs are defined in OpenAPI definition files, in YAML format. You can create a default OpenAPI definition file by using the **create** command and then modify it by using an editor of your choice.
- [Specifying a gateway type for an API definition](#)  
An API definition is specific to one or other of the gateway types, DataPower® API Gateway or DataPower Gateway (v5 compatible). API definitions must specify which type of gateway the API uses.
- [Referring to an extension in an API definition](#)  
To refer to an OpenAPI extension in your API definition YAML (or JSON) file, add an **extensions** key under **x-ibm-configuration**.
- [Using \\$ref to reuse code fragments in your OpenAPI files](#)  
If you deploy an API to an IBM API Connect Management server by using the developer toolkit command line, you can use the **\$ref** field in your OpenAPI YAML and JSON API definition files to reference a fragment of OpenAPI code that is defined in a separate file. When API Connect processes the source API definition file, the **\$ref** field is replaced with the contents of the target file.
- [Creating a Product definition file](#)  
You define a Product by creating a Product definition file.
- [Using x-ibm-languages to create multilingual API and Product documentation](#)  
Create multilingual API and Product documentation by using the **x-ibm-languages** extension in your API and Product OpenAPI definitions.
- [Using x-example to control the examples displayed in the Developer Portal](#)  
OpenAPI does not support the use of **example** attributes on parameters, only on request and response objects and their properties. To allow API Developers to be able to control the examples displayed in the portal you can use **x-example** in OpenAPI parameters.
- [Using x-embedded-doc to add additional documentation to products and APIs](#)  
You can use **x-embedded-doc** to add additional documentation to a product or an API as part of their definition.

- [Using x-pathaliases to give consistent URLs for products and APIs](#)  
You can use `x-pathaliases` to assign a URL to a product or an API.
- [Validating the YAML or JSON definition of an API or Product](#)  
You can validate a YAML or JSON definition by using the IBM API Connect developer toolkit.
- [Creating and using API and Product definitions templates](#)  
You can use template files when creating OpenAPI 2.0 and OpenAPI 3.0 APIs, and when creating Product definitions. Template files are Handlebars templates containing variables of the form `{{variable-name}}` that are substituted with values when you create the API or Product definition.

## Related reference

---

- [API development and management commands](#)

## Creating an OpenAPI definition file

APIs are defined in OpenAPI definition files, in YAML format. You can create a default OpenAPI definition file by using the `create` command and then modify it by using an editor of your choice.

You can stage or publish the API directly to a Catalog in API Manager by referencing the API in a Product definition file, and then using the `apic products:publish` command to publish the Product. You can also create a draft API in API Manager by using the `apic draft-apis:create` command.

You can create an API in the CLI by running `apic create:api`, and supplying additional arguments on the command line.

Another option is to create an API interactively in the command line by running `apic create:api` and following the prompts.

You can see further details and available options for the `apic create:api` command by running:

```
apic create:api --help
```

IBM provides an extension to the OpenAPI specification; this extension is described in [IBM extensions to the OpenAPI specification](#).

Note: Products that contain an API with a Swagger property using `regex` that include lookahead assertions, such as "(?" cannot be validated or published. An error message is returned. For example:

```
Product has not been published!
The multipart 'openapi' field contains an OpenAPI definition with validation errors.
 definitions.properties.pattern Does not match format 'regex' (context: (root).definitions.properties.pattern, line: 0, col:
0)
400
```

## Creating an API definition from a template

---

You can use a custom Handlebars template to create an API by using the following command:

```
apic create:api --template template_filename --title api_title
```

where `template_filename` is the name of the Handlebars template to use, and `api_title` is the title of your API.

An API template file must have a `.hbs` file name extension. You can create a template from scratch, or start with the example (default) API template provided in [API and Product definition template examples](#).

You can create multilingual API and Product documentation by using an `x-ibm-languages` extension directly in the OpenAPI definition. For more information, see [Using x-ibm-languages to create multilingual API and Product documentation](#).

- [IBM extensions to the OpenAPI specification](#)  
You use the `x-ibm-configuration` object in your OpenAPI definition file to add extensions that are specific to IBM API Connect.

## Related reference

---

- [API development and management commands](#)

## IBM extensions to the OpenAPI specification

You use the `x-ibm-configuration` object in your OpenAPI definition file to add extensions that are specific to IBM® API Connect.

The `x-ibm-configuration` extension has the following structure:

```
x-ibm-configuration:
 enforced: enforced_boolean
 phase: Phase
 testable: test_boolean
 cors:
 enabled: cors_boolean
 activity-log:
 activity_logging_extension
 assembly:
 execute:
 - assembly_component
 gateway: gateway_type
 type: api_type
```

```

properties:
 properties_extension
catalogs:
 catalogs_extension

```

The following table lists the various extensions used by API Connect, whether they are required, a description of their use and behavior, and their type.

Table 1. IBM extensions

Extension	Required	Description	Type
phase	Yes	Use the <b>phase</b> extension to describe the maturity of the API. It can take the following values: <ul style="list-style-type: none"> <li><b>identified</b>: the API is in the early conceptual phase and is neither fully designed nor implemented.</li> <li><b>specified</b>: the API has been fully designed and passed an internal milestone but has not yet been implemented.</li> <li><b>realized</b>: the API is in the implementation phase.</li> </ul>	String
testable	Yes	Use the <b>testable</b> extension to specify whether the API can be tested using the test tool in the Developer Portal.	Boolean
enforced	Yes	Use the <b>enforced</b> extension to specify if the API Connect gateway is used to enforce the API. <ul style="list-style-type: none"> <li><b>true</b> indicates that the API Connect gateway is used to enforce the API.</li> <li><b>false</b> indicates that the API Connect gateway is not used to enforce the API.</li> </ul>	Boolean
cors	Yes	Use the <b>cors</b> extension to specify whether CORS access control is used for the API. The extension has an <b>enabled</b> field which is a Boolean.	Object (with a single Boolean field)
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> activity-log	No	Use the <b>activity-log</b> extension to configure your logging preferences for the API activity that is stored in Analytics. The preferences that you specify will override the default settings for collecting and storing details of the API activity.	Object
assembly	No	Use the <b>assembly</b> extension to describe the application of policies and logic to an API. It contains an <b>execute</b> field that contains an array of policies that are applied in order. It can contain a <b>catch</b> field that contains an array of error cases to be caught. For information about use of the <b>execute</b> field, see <a href="#">execute</a> . For information about use of the <b>catch</b> field, see <a href="#">catch</a> .	Object
gateway	No	Use the <b>gateway</b> extension to specify which type of gateway you want to use. If you are using the DataPower® Gateway (v5 compatible) or DataPower API Gateway, it must be included and take one of the following values: <ul style="list-style-type: none"> <li><b>datapower-gateway</b> (DataPower Gateway (v5 compatible))</li> <li><b>datapower-api-gateway</b> (DataPower API Gateway)</li> </ul> For information about types of gateways, see <a href="#">API Connect gateway types</a> .	String
type	No	The <b>type</b> extension takes one of the following values: <ul style="list-style-type: none"> <li><b>rest</b></li> <li><b>wSDL</b></li> </ul>	
properties	No	Use the <b>properties</b> extension to define properties for use in an API.	Object ( <a href="#">properties</a> )
catalogs	No	Use the <b>catalogs</b> extension to define Catalog-specific values for properties defined in the <b>properties</b> extension.	Object ( <a href="#">catalogs</a> )

The following example shows the **x-ibm-extension** section of an API that is enforced by API Connect, is in the realized state, is testable through the test tool in the Developer Portal, has CORS access control enabled, and has a simple assembly that invokes a URL and then redacts a field from the request or response.

```

x-ibm-configuration:
 enforced: true
 phase: realized
 testable: true
 cors:
 enabled: true
 assembly:
 execute:
 - invoke:
 title: Example Invoke
 target-url: 'https://example.com/api'
 description: Example description
 - redact:
 actions:
 - action: redact
 from:
 - request
 - response
 path: /**[@name='secondaryAddress']/*[@name='streetAddress']
 properties:
 ID:
 value: 1234
 description: An ID to be used for validating.
 encoded: false
 catalogs:
 Sandbox:
 properties:
 ID: 5678

```

- [catch](#)  
Use the **catch** extension to catch errors that occur during an API call.
- [properties](#)  
Use the **properties** extension to define properties for referencing in an API.



- **catalogs**  
Use the **catalogs** extension to assign catalog-specific values to properties defined in the **properties** section.
- **activity-log**  
If you're using the DataPower API Gateway, you can use the **activity-log** extension to configure your logging preferences for the API activity that is stored in Analytics. The preferences that you specify will override the default settings for collecting and storing details of the API activity.

## execute

The execute field of an assembly has the following structure:

```
execute:
- Policy_1
- Policy_2
```

Note: Although some built-in policies can be used with both the DataPower® Gateway (v5 compatible) and the DataPower API Gateway, some policies are restricted to a particular Gateway. The following icons indicate which Gateway each policy can be used with:

-  Indicates that the policy can be run on the DataPower Gateway (v5 compatible).
-  Indicates that the policy can be run on the DataPower API Gateway.

For details of the two types of gateway, see [API Connect gateway types](#).

The following table describes the possible policies and logic constructs that can be included in an execute field.

Table 1. execute properties

Property	Required	Description	Data Type		
activity-log	No	Use the activity-log policy to log information that relates to the calling of API operations.	object ( <a href="#">activity-log</a> )	✓	✓ Functionality provided by the <a href="#">activity-log</a> extension
client-security	No	Provides a range of options for authenticating client access to your APIs, extending the capabilities of the OpenAPI specification.	object ( <a href="#">client-security</a> )	✗	✓
gatewayscript	No	Include a GatewayScript program.	object ( <a href="#">gatewayscript</a> )	✓	✓
graphql-introspect	No	Use the graphql-introspect policy to introspect a GraphQL schema.	object ( <a href="#">graphql-introspect</a> )	✗	✓
if	No	Use the if policy to execute a section of the assembly only when a condition is fulfilled.	object ( <a href="#">if</a> )	✓	✓ Functionality provided by <a href="#">switch</a>
invoke	No	Use the invoke policy to call an API. The last invoke in your policy might be replaced by a proxy automatically to improve performance. To disable this, see: <a href="#">API properties</a> .	object ( <a href="#">invoke</a> )	✓	✓
json-to-xml	No	Convert payload from JSON to XML.	object ( <a href="#">json-to-xml</a> )	✓	✓
jwt-generate	No	Generate a JSON Web Token (JWT).	object ( <a href="#">jwt-generate</a> )	✓	✓
jwt-validate	No	Validate a JSON Web Token (JWT).	object ( <a href="#">jwt-validate</a> )	✓	✓
log	No	Use the log policy to customize or override the default activity logging configuration for an API.	object ( <a href="#">log</a> )	✗	✓
map	No	Use the map policy to transform variables.	object ( <a href="#">map</a> )	✓	✓
operation-switch	No	Use the operation-switch policy when you want to execute alternative policy assemblies, conditional on the operation that is being called.	object ( <a href="#">operation-switch</a> )	✓	✓
oauth	No	Use the oauth policy to perform OAuth processing based on defined OAuth provider settings.	object ( <a href="#">oauth</a> )	✗	✓
parse	No	Use the parse policy to control the parsing of an input document. When the input document is a JSON string, the string is parsed instead of copied over.	object ( <a href="#">parse</a> )	✗	✓
proxy	No	Proxy a service.	object ( <a href="#">proxy</a> )	✓	✓ Functionality provided by <a href="#">invoke</a>
ratelimit	No	Use the ratelimit policy to apply one or more rate or burst limits at any point in your API assembly flow. Rate and burst limits restrict the number of calls that an application can make to an API in a specified time period.	object ( <a href="#">ratelimit</a> )	✗	✓
redact	No	Use the redact policy to completely remove or to redact specified fields from the request body, the response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.	object ( <a href="#">redact - DataPower API Gateway</a> <a href="#">redact - DataPower Gateway (v5 compatible)</a> )	✓	✓
set-variable	No	Use the set-variable policy to set a runtime variable to a string value, or to add or clear a runtime variable.	object ( <a href="#">set-variable</a> )	✓	✓
switch	No	Use the switch policy to execute one of a number of sections of the assembly based on which specified condition is fulfilled.	object ( <a href="#">switch</a> )	✓	✓

Property	Required	Description	Data Type	DataPower Gateway	API Gateway
throw	No	Use the throw policy to specify points at which an error should be thrown.	object ( <a href="#">throw</a> )	✓	✓
user-defined-policy	No	You can apply your own user-defined policies to your APIs.	object ( <a href="#">User-defined policies</a> )	✓	✓
user-security	No	Extract a user's credentials, authenticate those credentials, and obtain authorization from the user.	object ( <a href="#">user-security</a> )	✗	✓
validate	No	Use the Validate policy to validate the payload in an assembly flow against a JSON or an XML schema.	object ( <a href="#">validate - DataPower API Gateway</a> , <a href="#">validate - DataPower Gateway (v5 compatible)</a> )	✓	✓
validate-username-token	No	Validate a WS-Security UsernameToken.	object ( <a href="#">validate-username-token</a> )	✓	✗
xml-to-json	No	Convert payload from XML to JSON.	object ( <a href="#">xml-to-json</a> )	✓	✓
xslt	No	Apply an XSLT transform to the payload.	object ( <a href="#">xslt</a> )	✓	✓

The following example shows an execute field for an assembly that invokes a URL and then redacts a field from the request or response.

```
execute:
- invoke:
 title: Example Invoke
 target-url: 'https://example.com/api'
 description: Example description
- redact:
 actions:
 - action: redact
 from:
 - request
 - response
 path: /**[@name='secondaryAddress']/**[@name='streetAddress']
```

## activity-log

Use the Activity Log policy to configure your logging preferences for the API activity that is stored in IBM® API Connect analytics. The preferences that you specify will override the default settings for collecting and storing details of the API activity.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Activity Log](#).

Note: If you are using the DataPower API Gateway, you configure your logging preferences by using the [activity-log](#) extension.

## About

The activity-log policy has the following format:

```
- activity-log:
 version: version
 title: title
 description: description
 content: activity_to_log_if_call_successful
 error-content: activity_to_log_if_call_unsuccessful
```

Apply this policy by adding an assembly extension with an execute field to your OpenAPI definition file.

You can also apply an activity-log policy by using the API Designer assembly editor to add a built-in policy to the API. For more information, see [Activity Log](#) in the built-in policies section.

Note: If payload logging is enabled, for the gateway to capture payloads buffering must also be enabled.

```
activity-log:
 success-content: activity_to_log_if_call_successful
 error-content: activity_to_log_if_call_unsuccessful
 enabled: is_activity_logging_enabled
 buffering: true
```

## Properties

The following table describes the policy properties:

Table 2. Activity Log policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	Yes	A title for the policy.	string

Property	Required	Description	Data type
description	No	A description of the policy.	string
content	Yes	<p>Defines the type of content to be logged when the operation is successful.</p> <p>Valid values:</p> <ul style="list-style-type: none"> <li><b>none</b>: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal.</li> <li><b>activity</b>: Logs invocation only (only the resource URI is recorded).</li> <li><b>header</b>: Logs activity and header.</li> <li><b>payload</b>: Logs activity, header, and payload (the original request, if any, and the final response).</li> </ul> <p>The default value is <b>activity</b>.</p>	string
error-content	No	<p>Indicates what content to log if an error occurs.</p> <p>Valid values:</p> <ul style="list-style-type: none"> <li><b>none</b>: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal.</li> <li><b>activity</b>: Logs invocation only (only the resource URI is recorded).</li> <li><b>header</b>: Logs activity and header.</li> <li><b>payload</b>: Logs activity, header, and payload (the original request, if any, and the final response).</li> </ul> <p>The default value is <b>payload</b>.</p>	string

## Example 1

```
use defaults
- activity-log:
 version: 1.0.0
 title: default activity logging
```

## Example 2

```
- activity-log:
 version: 1.0.0
 title: no logging for successful calls
 content: none
 error-content: activity
```

## client-security

The client-security policy can be used to extend client authentication access for your APIs.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Client Security](#).

## About

You use the client-security policy to create an assembly action that's separate from the OpenAPI specification, to allow for more options to authenticate an application. Note that you cannot use the client-security policy to perform rate-limiting enforcement.

In a client security action, you define the following settings.

- Control whether to stop processing if client security fails. When client security fails, stops assembly processing and returns an error.
- Control whether to require the client secret. When required, the secret is compared to the registered secret on the application that is identified by the client ID.
- Define the method to extract client credentials from the request.
  - For all methods except **http**, use the **id-name** and the **secret-name** properties to specify the location that contains the ID and the location that contains the secret.
    - When **cookie**, specify which cookie.
    - When **context-var**, specify which runtime context variable.
    - When **form**, specify the form data.
    - When **header**, specify which header.
    - When **query**, specify which query parameter.
  - For the **http** method, use the **http-type** property to specify the format of the authorization header, which expects the **basic** form in the **basic base64 id:secret** format.
- Define the method to authenticate the extracted client credentials. The supported methods are **native** (this means by using IBM® API Connect), or by using a specified **third-party** user registry.
  - If **third-party**, use the **user-registry** property to specify the user-registry to authenticate the extracted client credentials. The supported registry types are LDAP and authentication URL.

## Properties

The client-security policy has the following format:

```
- client-security:
 version: version
 title: title
 description: description
 stop-on-error: is_processing_stopped_on_client_security_failure
 secret-required: is_client_secret_required_in_request
 extract-credential-method: method_for_supplying_credentials
 id-name: parameter_that_specifies_client_id
 secret-name: parameter_that_specifies_client_secret
 http-type: authentication_type
 client-auth-method: method_for_client_authentication
 user-registry: user_registry_for_client_authentication
```

Table 2. client-security policy properties

Property	Required	Description	Data type
version	Yes	The policy version number.	string
title	No	The title of the policy.	string
description	No	A description of the policy.	string
stop-on-error	Yes	If set to <b>true</b> , assembly processing stops if client security fails, and an error is returned.	boolean
secret-required	Yes	If set to <b>true</b> , the client secret must be sent in the request. The secret is compared to the registered secret on the application that is identified by the client ID.	boolean
extract-credential-method	Yes	Specify one of the following values to define how the calling application authenticates: <ul style="list-style-type: none"> <li>• <b>header</b>: client ID and client secret credentials must be supplied in the request header.</li> <li>• <b>query</b>: client ID and client secret credentials must be supplied as query parameters in the request URL.</li> <li>• <b>form</b>: client ID and client secret credentials must be supplied as form data sent in a POST request.</li> <li>• <b>cookie</b>: client ID and client secret credentials must be supplied in a header called <b>Cookie</b>.</li> <li>• <b>http</b>: the calling application must authenticate by using basic authentication.</li> <li>• <b>context-var</b>: the credentials that are used for authenticating the client are obtained from context variables that you set in the assembly flow prior to the Client Security policy, by using a <a href="#">gatewayscript</a> policy for example. The names of these context variables are determined by the values that you supply in the id-name and secret-name properties of the client-security policy.</li> </ul>	string
id-name	Yes, unless extract-credential-method is set to <b>http</b>	The name of the parameter whose value specifies the client ID. For all values of extract-credential-method other than <b>context-var</b> and <b>http</b> , the calling application must supply a parameter with this name, in the location defined by the extract-credential-method setting. For <b>context-var</b> , this property specifies the name of a context variable. This option does not apply if the extract-credential-method property is set to <b>http</b> .	string
secret-name	Yes, if secret-required is set to <b>true</b> and extract-credential-method is other than <b>http</b>	The name of the parameter whose value specifies the client secret. For all values of extract-credential-method other than <b>context-var</b> and <b>http</b> , the calling application must supply a parameter with this name, in the location defined by the extract-credential-method setting. For <b>context-var</b> , this property specifies the name of a context variable. This option does not apply if the extract-credential-method property is set to <b>http</b> .	string
http-type	Yes, if extract-credential-method is set to <b>http</b>	The authentication type. Currently, this must be set to <b>basic</b> .	string
client-auth-method	Yes	Specify one of the following values: <ul style="list-style-type: none"> <li>• <b>native</b>: only client ID and client secret are used to authenticate the request. If extract-credential-method is set to <b>http</b> then the calling application must supply the client ID for the user name, and the client secret for the password.</li> <li>• <b>third-party</b>: a user registry is used to authenticate the client. If extract-credential-method is set to a value other than <b>http</b> then the calling application must supply the user name for the client ID, and the password for the client secret.</li> </ul>	string
user-registry	Yes, if the client-auth-method is set to <b>third-party</b>	Specify the value of the <b>name</b> property of the user registry that will be used to authenticate the client. The supported registry types are LDAP and authentication URL.	string

## client-security policy example

```
- client-security:
 version: 2.0.0
 title: client-security
 stop-on-error: true
 secret-required: true
 extract-credential-method: cookie
 id-name: my-client-id
 secret-name: my-client-secret
 client-auth-method: third-party
 user-registry: myauthurl
```

API Gateway only



---

## extract

Use the extract policy to extract and transform data from fields in the API context.

## Gateway support

---

Table 1. Gateways that support this policy and the corresponding policy versions

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Extract](#).

## About

---

The extract policy specifies the data source that contains the content to transform, the fields that contain the content, and expressions that define how to transform the content. You use a subset of JSONata notation to specify the fields to extract and transform. For more information, see [Constructing JSONata expressions to extract and transform data](#).

The input to the extract policy must be parsed data. One way to produce parsed data is to add a parse policy before the extract policy in your assembly.

The extract policy has the following format:

```
- extract:
 version: 2.0.0
 title: title
 root: data source
 extracts:
 - capture: path expression that identifies the field
 transform: expression that defines how to transform the content
 .
 .
 .
 description: description
```

Apply this policy by adding an `assembly` extension with an `execute` field to your OpenAPI definition file.

## Properties

---

Table 2. extract policy properties

Property label	Required	Description	Data type
version	Yes	The policy version number. You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.	string
title	Yes	The title of the policy. The default value is <code>extract</code> .	string
description	No	A description of the policy.	string
root	Yes	The data source that contains the content to transform. The default value is <code>message.body</code> .	string
capture	Yes	The path expression that identifies the field. The default setting is <code>\$</code> , which indicates the entire input.	string
transform	Yes	The expression that defines how to transform the content.	string

## Example

---

Transform the contents of the `account` field to include only the last 4 characters.

```
- extract:
 version: 2.0.0
 title: extract
 root: message.body
 extracts:
 - capture: $.members.policy.**.account
 transform: $substring($,-4)
 description: Include only the last 4 characters of the account field.
```

---

## gatewayscript

Use the gatewayscript policy to execute a specified DataPower GatewayScript program.

## Gateway support

---

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [GatewayScript](#).

## About

The gatewayscript policy has the following structure:

```
- gatewayscript:
 version: version
 title: Title
 description: Description
 source: Script
```

## Properties

The following table describes the policy properties:

Table 2. gatewayscript policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	The title of the policy.	string
description	No	A description of the policy.	string
source	Yes	The GatewayScript source code to execute.	string

## Example

The following is an example of a simple gatewayscript policy:

```
- gatewayscript:
 version: 1.0.0
 title: Example_GatewayScript
 source: console.debug('Hello World!');
 description: A simple GatewayScript policy.
```

For more information about how to use a gatewayscript policy, see [GatewayScript](#) in the built-in policies section.

For more code examples, see [GatewayScript code examples](#) and, if you are using the DataPower API Gateway, [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway](#).

For general information on using GatewayScript, see the following topics in the DataPower product documentation:

- [GatewayScript APIs for API management](#)
- [Accessing and manipulating a variable in the API context](#)
- [OAuth context variables](#)

## Related concepts

- [Variable references in API Connect](#)

## Related reference

- [GatewayScript code examples](#)
- [API Connect context variables](#)

## graphql-introspect

Use the graphql-introspect policy to introspect a GraphQL schema.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [GraphQL introspect](#).

## Properties

Table 2. graphql-introspect policy properties

Property	Required	Description	Data type
version	Yes	The policy version number.	string
title	No	The title of the policy.	string
description	No	A description of the policy.	string
input	No	A variable in the API context that contains the input to introspect. The content of the <b>body</b> field of the variable is the input to introspect. By default, the variable name is <b>message</b> .	string
output	No	A variable in the API context where the results of the introspection are stored. The content of the <b>body</b> field of the variable is the result of GraphQL introspection. The default variable name is the same as that of the variable specified for the <b>input</b> field. Therefore, by default, the input of the introspection is overwritten by the output.	string

## Example

```
- graphql-introspect:
 version: 2.0.0
 title: introspect the accounts graphql schema
 input: message
```

## if

Use the if construct when you want to execute a portion of your assembly only when a specific condition is fulfilled.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway, functionality provided by <a href="#">switch</a>	

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [if](#).

## About

The if policy has the following format:

```
- if:
 version: version
 title: title
 description: description
 condition: 'condition_1'
 execute:
 policy_assembly_1 ...
```

In the condition field, use the form `apim.getvariable('context.location.variable')` to reference your variables, where *context* is the context that you want to reference, *location* is the location of the variable within that context, and *variable* is the name of the variable.

The **execute**: section can define any policy assembly, including further if policies. For more information, see [execute](#).

## Properties

Table 2. if policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
condition	Yes	A script that returns <b>true</b> or <b>false</b> . Use GatewayScript for a DataPower Gateway implementation. For information about the context variables you can use and how to reference them in your script, see <a href="#">API Connect context variables</a> .	string
execute	Yes	The policy assembly that you want to execute if the condition returns <b>true</b> . For more information, see <a href="#">execute</a> .	string

## Example

```
carry out different redaction actions depending on the operation

- if:
 version: 1.0.0
 title: clear_region_and_set_body
 condition: 'apim.getvariable('request.body.secret') == true'
 execute:
 - redact:
 title: remove secret field
 actions:
 - action: remove
 from: all
 path: /document/user/secret
```

# invoke

Use the invoke policy to call an API.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.1.1 or later) 2.2.0 (DataPower API Gateway Version 10.0.2.0 or later) 2.3.0 (DataPower API Gateway Version 10.0.3.0 or later)

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Invoke](#).

## About

The invoke policy does not support responses with multipart form data, that is, when the response is set to Content-Type: multipart/related.

The invoke policy has the following format:

```
- invoke:
 version: version
 title: title
 description: description
 target-url: URL_of_target_API
 backend-type: how_payload_is_sent_to_backend
 tls-profile: TLS_profile_to_be_used
 verb: method_type
 timeout: timeout_value_in_seconds
 compression: is_data_to_be_compressed
 username: username_if_authentication_required
 password: password_if_authentication_required
 output: location_of_the_invoke_result
 cache-key: unique_identifier_of_the_document_cache_entry
 cache-response: cache_behavior
 cache-putpost-response: response_caching_behavior
 cache-ttl: cache_time_to_live
 inject-proxy-headers: are_proxy_headers_sent_to_target_url
 decode-request-params: are_request_parameters_decoded
 encode-plus-char: are_plus_characters_encoded
 keep-payload: is_payload_sent_on_delete
 use-http-10: are_transactions_restricted_to_http_1.0
 chunked-uploads: are_chunked_encoded_documents_sent_to_the_server
 persistent-connection: are_persistent_connections_enabled
 header-control:
 .
 .
 .
 headers_to_copy_to_target_url
 .
 .
 parameter-control:
 .
 .
 .
 parameters_to_copy_to_target_url
 .
 .
 follow-redirects: url_redirection_behavior
 stop-on-error: errors_that_stop_the_flow
```

## Properties

The following table describes the properties of the invoke policy.

Table 2. Invoke policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
target-url	Yes	The URL of the target API.	string

Property	Required	Description	Data type
<b>API Gateway only</b> backend-type	No	Specify one of the following values to determine how the payload is sent to the backend: <ul style="list-style-type: none"> <li><b>detect</b>: detect the message type and send the payload to the backend accordingly.</li> <li><b>xml</b>: package the payload as XML and send. If the message type is not XML, the operation fails.</li> <li><b>json</b>: package the payload as JSON and send. If the message type is not JSON, the operation fails.</li> <li><b>binary</b>: package the payload as binary and send, regardless of the message type.</li> <li><b>graphql</b>: package the payload as GraphQL and send. If the message type is not GraphQL, the operation fails.</li> </ul> <p>The default value is <b>detect</b>.</p>	string
<b>API Gateway only</b> graphql-send-type(policy version 2.2.0 and later)	Yes, if backend-type is set to <b>graphql</b> or <b>detect</b> , and verb is set to <b>POST</b> or <b>keep</b> .	Specify one of the following values to determine how the GraphQL payload is sent to the back end: <ul style="list-style-type: none"> <li><b>detect</b>: detect the message type and send the payload to the back end accordingly.</li> <li><b>graphql</b>: package the payload as GraphQL and send. If the message type is not GraphQL, the operation fails.</li> <li><b>json</b>: package the payload as JSON and send. If the message type is not GraphQL, the operation fails.</li> </ul> <p>Note: graphql-send-type is supported only if backend-type is set to <b>graphql</b> or <b>detect</b>, and verb is set to <b>POST</b> or <b>keep</b>.</p>	string
tls-profile	No	Specifies a TLS profile to use for the secure transmission of data.	string
verb	No	The operation method type.  Valid values: <ul style="list-style-type: none"> <li>GET</li> <li>POST</li> <li>PUT</li> <li>DELETE</li> <li>PATCH</li> <li>HEAD</li> <li>OPTIONS</li> </ul> <p>The default value is <b>GET</b>. However, if the property is omitted from the source, the HTTP method from the incoming request is used.</p>	string
timeout	No	The time to wait before a reply back from the endpoint (in seconds). The default value is <b>60</b> .	integer
<b>API Gateway only</b> http-version (policy version 2.1.0 and later)	Yes	The HTTP version that will be used when connecting to the backend server. Valid values: <ul style="list-style-type: none"> <li><b>HTTP/1.0</b></li> <li><b>HTTP/1.1</b></li> <li><b>HTTP/2</b></li> </ul>	string
<b>API Gateway only</b> http2-required (policy version 2.1.0 and later)	No	Set this property to <b>true</b> to require that the server selects HTTP/2 during a TLS connection, otherwise the DataPower API Gateway rejects the connection and fails the request. For a non TLS connection, a warning is logged stating that the requirement can't be enforced. This setting applies only if http-version is set to <b>HTTP/2</b> .	boolean
compression	No	Specifies whether data is to be compressed by using compress before it is uploaded. The default value is <b>false</b> .	boolean
username	No	The username to use for HTTP Basic authentication.	string
password	No	The password to use for HTTP Basic authentication.	string
output	No	The name of a variable that will be used to store the response data from the request. By default, the invoke response, that is the body, headers, statusCode and statusMessage, is saved in the variable <i>message</i> . Use this property to specify an alternate location to store the invoke response. This variable can then be referenced in other actions, such as <a href="#">map</a> . Note: If you want the response to be saved in <i>message</i> , leave the output property blank, do <b>not</b> supply the value <i>message</i> .	string
cache-key	No	Specifies the unique identifier of the document cache entry.	string
<b>API Gateway only</b> websocket-upgrade (policy version 2.1.0 and later)	No	Set this property to <b>true</b> to allow an HTTP or HTTPS connection to be upgraded to the WebSocket protocol if the request received by the corresponding HTTP or HTTPS handler on the DataPower API Gateway uses WebSocket (ws) or WebSocket Secure (wss). Note: <ul style="list-style-type: none"> <li>This property is available only for GraphQL APIs.</li> <li>The <b>websocket-upgrade</b> property is deprecated. To allow WebSocket upgrade requests that can manage API processing data, define an assembly WebSocket upgrade policy. For more information, see <a href="#">websocket-upgrade</a>.</li> </ul>	boolean
cache-response	No	The cache response type. Valid values: <ul style="list-style-type: none"> <li><b>protocol</b>: The cache behavior is defined by the Cache-Control headers on the request and response.</li> <li><b>no-cache</b>: Specifies that there is no caching. However, if the document is already in the cache, the document is retrieved from the cache.</li> <li><b>time-to-live</b>: Specifies that the response stays in the cache for the specified time.</li> </ul> <p>The default value is <b>protocol</b>.</p>	string

Property	Required	Description	Data type
cache-putpost-response	No	Specifies whether to cache the response from POST and PUT requests. Caching the response from POST and PUT requests can reduce server load and reduce latency in the response to the client request. The default value is <b>false</b> .	boolean
cache-ttl	No	Specifies the amount of time in seconds that the response stays in the cache. Applies only if the property cache-response is set to <b>time-to-live</b> . Enter a value in the range 5 - 31708800. The default value is <b>900</b> .	integer
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> inject-proxy-headers	No	When set to true, the <b>invoke</b> policy injects the <b>X-Forwarded-For</b> , <b>X-Forwarded-To</b> , <b>X-Forwarded-Host</b> , and <b>X-Forwarded-Proto</b> headers to the request that is sent to the <b>target-url</b> . The default value is <b>false</b> .	boolean
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> decode-request-params	No	When set to true, any request parameters that are referenced by a variable definition on the <b>target-url</b> of the invoke policy are URL-decoded. The default value is <b>false</b> .	boolean
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> encode-plus-char	No	When set to true, all "+" characters in the query parameter values of the <b>target-url</b> are encoded to %2F. The default value is <b>false</b> .	boolean
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> keep-payload	No	When set to true, the invoke policy sends a payload on an <b>HTTP DELETE</b> method. The default value is <b>false</b> .	boolean
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> use-http-10 (policy version 2.0.0 only)	No	When set to true, HTTP transactions are restricted to version 1.0. The default value is <b>false</b> .	boolean
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> chunked-uploads	No	<p>If you set this property to <b>true</b>, chunked-encoded documents are sent to the server. When the HTTP 1.1 protocol is used, the body of the document can be delimited by either <b>Content-Length</b> or chunked encoding. While all servers can interpret <b>Content-Length</b>, many applications fail to understand chunked-encoded documents. For this reason, <b>Content-Length</b> is the standard method. The use of <b>Content-Length</b> interferes with the ability of the DataPower Gateway to fully stream. If you must stream full documents to the target server, enable this property.</p> <p>When enabled, the server must be RFC 2616 compatible. Unlike all other HTTP 1.1 features that can be negotiated down at run time, you must know beforehand that the target server is RFC 2616 compatible.</p> <p>The default value is <b>true</b>.</p> <p>Note: Chunked encoding is not supported by the HTTP 1.0 protocol.</p>	boolean
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> persistent-connection (policy version 2.3.0 and later)	No	Set this property to <b>true</b> to enable HTTP persistent connections, allowing connection re-use. The default value is <b>true</b> .	boolean
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> header-control: type values	No	<p>Specifies the headers in <b>message.headers</b> that you want to copy to the target URL. If the <b>type</b> property is set to <b>blocklist</b>, the <b>values</b> property lists the headers that you don't want to be copied. If the <b>values</b> property is empty then all headers are copied.</p> <p>If the <b>type</b> property is set to <b>allowlist</b>, the <b>values</b> property lists the headers that you want to be copied.</p> <p>The items that are listed in the <b>values</b> property are in regular expression format. The values are not case-sensitive. The default value of the <b>header-control</b> property is</p> <pre>header-control:   type: blocklist   values: []</pre> <p>See <a href="#">header-control examples</a></p>	object
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> parameter-control: type values	No	<p>Specifies the parameters in the incoming request that you want to be copied to the target URL. If the <b>type</b> property is set to <b>blocklist</b>, the <b>values</b> property lists the parameters that you don't want to be copied.</p> <p>If the <b>type</b> property is set to <b>allowlist</b>, the <b>values</b> property lists the parameters that you want to be copied. If the <b>values</b> property is empty then no parameters are copied.</p> <p>The items that are listed in the <b>values</b> property are in regular expression format. The values are not case-sensitive. The default value of the <b>parameter-control</b> property is</p> <pre>parameter-control:   type: allowlist   values: []</pre> <p>See <a href="#">parameter-control examples</a></p>	object
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> follow-redirect	No	Specifies the behavior if the back-end server returns the HTTP status code <b>301 Moved Permanently</b> . If this property is set to <b>true</b> , the <b>invoke</b> policy follows the URL redirection by making a further call to the URL specified in the <b>Location</b> header in the response. If this property is set to <b>false</b> , the <b>invoke</b> saves the <b>301</b> status code and the API call is considered to be complete. Note: The <b>follow-redirect</b> property is supported only by the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), the <b>invoke</b> always follows the URL redirection; the <b>proxy</b> policy (not supported by the DataPower API Gateway) saves the <b>301</b> status code and completes the API call without following the URL redirection.	boolean

Property	Required	Description	Data type
stop-on-error	No	List the errors that, if thrown during the policy execution, cause the flow to stop. If there is a <code>catch</code> flow configured for the error, it is triggered to handle the error thrown. If an error is thrown and there are no errors specified for the Stop on error property, or if the error thrown is not one of the specified errors, the policy execution is allowed to complete, and the assembly flow continues.	string

## Example

```
- invoke:
 version: 2.0.0
 title: get the account status
 target-url: https://example.com/accounts/{id}?status={status}
 cache-response: time-to-live
 cache-putpost-response: true
 tls-profile: MyTLSProfile
 verb: POST
 timeout: 60
 compression: false
 username: MyUser
 password: MyPassword
 stop-on-error:
 - ConnectionError
 - OperationError
```

API Gateway only

## header-control examples

```
copy all headers to the target URL

- invoke:
 target-url: http://myhost/mypath
 header-control:
 type: blocklist
 values: []

copy all headers except X-Client-ID and Content-Type

- invoke:
 target-url: http://myhost/mypath
 header-control:
 type: blocklist
 values:
 - ^X-Client-ID$
 - ^Content-Type$

copy no headers

- invoke:
 target-url: http://myhost/mypath
 header-control:
 type: allowlist
 values: []

copy only the Content-Type header

- invoke:
 target-url: http://myhost/mypath
 header-control:
 type: allowlist
 values:
 - ^Content-Type$
```

API Gateway only

## parameter-control examples

```
copy no request parameters to the target URL

- invoke:
 target-url: http://myhost/path?storeid=3
 parameter-control:
 type: allowlist
 values: []

append the petid parameter to the target URL
if the incoming request is http://apigw/org/sandbox/petstore/base?petid=100&display=detailed,
the target URL at runtime will be http://myhost/mypath?storeid=3&petid=100

- invoke:
 target-url: http://myhost/path?storeid=3
 parameter-control:
 type: allowlist
 values:
 - ^petid$
```

## json-to-xml

Use the json-to-xml policy to convert the context payload of your API from the JavaScript Object Notation (JSON) format to the extensible markup language (XML) format.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [JSON to XML](#).

## About

The json-to-xml policy has the following structure:

```
- json-to-xml:
 version: version
 title: Title
 description: Description
```

Note: If you are using the DataPower API Gateway, the input to the json-to-xml policy must be parsed data. One way to produce parsed data is to use a [parse](#) policy before a json-to-xml policy in your assembly flow, which provides explicit control of the parse action.

## Properties

The following table describes the policy properties:

Table 2. Policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	Yes	The title of the policy.	string
description	No	A description of the policy.	string
<span>API Gateway only</span> input	No	The input message to convert. Specify the name of a variable in the API context. <b>variableName.body</b> , the message payload, represents the JSON input to convert. The default value of the variable is <b>message</b> and <b>message.body</b> is the default input.	string
<span>API Gateway only</span> output	No	The output message to store the conversion result. Specify the name of a variable in the API context. <b>variableName.body</b> represents the result of conversion from JSON format to XML format. When the specified input message is the default message, the default output is <b>message.body</b> . Otherwise, when the input message is the variable <b>my-message-variable</b> , for example, the default output is <b>my-message-variable.body</b> . The variable cannot be any read-only in the API context.	string
<span>API Gateway only</span> conversionType	No	The conversion type that determines the target format of the output. The following options are available: <ul style="list-style-type: none"><li>None: No conversion of the output takes place.</li><li>badgerFish: BadgerFish convention is used to determine the target conversion format of the output.</li></ul>	string
root-element-name	Yes	The root element name of the resultant XML document. This property is used only if the input JSON document is not hierarchical and has more than one uppermost level property, or if the always-output-root-element property is set to <b>true</b> .	
always-output-root-element	Yes	Select this property to <b>true</b> you always want the policy to output the root element, even if it is not required to make the XML document well formed.	boolean
unnamed-element-name	No	The XML element name to be used for JSON array elements.	string

## Example

The following is an example of a json-to-xml policy:

```
- json-to-xml:
 version: 1.0.0
 title: JSON to XML transform
 description: Transforms JSON message body to XML format
```

## jwt-generate

Use the Generate JWT security policy in IBM® API Connect to generate a JSON Web Token (JWT).

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Generate JWT](#).



## About

The jwt-generate policy has the following structure:

```
- jwt-generate:
 version: version
 title: title
 description: description
 jwt: json_web_token
 jti-claim: jwt_id_claim
 iss-claim: issuer_claim
 exp-claim: validity_period
 sub-claim: subject_claim
 aud-claim: audience_claim
 jws-jwk: sign_jwk_variable_name
 jws-alg: cryptographic_algorithm
 jws-crypto: sign_crypto_object
 jwe-enc: encryption_algorithm
 jwe-jwk: encrypt_jwk_variable_name
 jwe-alg: key_encryption_algorithm
 jwe-crypto: encrypt_crypto_object
```

## Properties

The following table describes the policy properties:

Table 2. Generate JWT policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	The title of the policy.	string
description	No	A description of the policy.	string
jwt	No	Runtime variable in which to place the JWT that is generated. The default value is: <b>generated.jwt</b> . However, if not set, the JWT that is generated is written to the Authorization Header as a Bearer token.	string
jti-claim	No	Indicates whether a JWT ID (jti) claim should be added to the JWT. If selected, the property is set to <b>true</b> , and a UUID is generated and set as the JTI claim value.	boolean
iss-claim	Yes	Runtime variable from which the Issuer (iss) claim string can be retrieved. This claim represents the Principal that issued the JWT. The default value is: <b>iss.claim</b>	string
sub-claim	No	Runtime variable from which the Subject (sub) claim string can be retrieved.	string
aud-claim	No	Runtime variable from which the Audience (aud) claim string can be retrieved. Multiple variables are set by using a comma-separated string.	string
exp-claim	Yes	The length of time (in seconds), that is added to the current date and time, in which the JWT is considered valid. The default value is <b>3600</b> .	integer
private-claims	No	Runtime variable from which a valid set of JSON claims can be retrieved. These claims are added to any set of claims specified previously.	string
jws-jwk	No	Runtime variable that contains the JWK that is used to sign the JWT. A JWK and a Crypto Object are both valid ways of providing the cryptographic data necessary to sign the JWT. However, if both data types are specified, only the Crypto Object is used.	string
jws-alg	No	The cryptographic algorithm to use. Valid values are: <ul style="list-style-type: none"><li>• HS256</li><li>• HS384</li><li>• HS512</li><li>• RS256</li><li>• RS384</li><li>• RS512</li><li>• ES256</li><li>• ES384</li><li>• ES512</li><li>• PS256</li><li>• PS384</li><li>• PS512</li></ul>	string
jws-crypto	No	The cryptographic object to use to sign the JWT. A JWK and a Crypto Object are both valid ways of providing the cryptographic data necessary to sign the JWT. However, if both data types are specified, only the Crypto Object is used.	string
jwe-enc	No	The encryption algorithm to use. Valid values are: <ul style="list-style-type: none"><li>• A128CBC-HS256</li><li>• A192CBC-HS384</li><li>• A256CBC-HS512</li></ul>	string
jwe-jwk	No	Runtime variable that contains the JWK to use to encrypt the JWT.	string

Property	Required	Description	Data type
jwe-alg	No	The key encryption algorithm to use. Valid values are: <ul style="list-style-type: none"> <li>• RSA1_5</li> <li>• RSA-OAEP</li> <li>• RSA-OAEP-256</li> <li>• dir</li> <li>• A128KW</li> <li>• A192KW</li> <li>• A256KW</li> </ul>	string
jwe-crypto	No	The cryptographic object to use to encrypt the claim.	string

## Example

The following is an example of a jwt-generate policy:

```
- jwt-generate:
 version: 1.0.0
 title: jwt-generate
 iss-claim: iss.claim
 exp-claim: 3600
 jwt: generated.jwt
 jti-claim: true
 sub-claim: sub.claim
 aud-claim: aud.claim
 private-claims: private.claims
 jws-jwk: jws.jwk
 jws-alg: HS256
 jws-crypto: jwsCryptoObjectName
 jwe-enc: A128CBC-HS256
 jwe-jwk: jwe.jwk
 jwe-alg: A128KW
 jwe-crypto: jweCryptoObjectName
```

For more information about how to use a jwt-generate security policy, see [Generate JWT](#) in the built-in policies section.

## jwt-validate

Use the Validate JWT security policy to enable the validation of a JSON Web Token (JWT) in a request before allowing access to the APIs.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Validate JWT](#).

Note:

- If the original message was signed with a Shared Secret Key, the cryptographic object that is specified must also be a Shared Secret Key.
- If the original message was signed with a Private Key, the cryptographic object that is specified must be a Crypto Certificate (public certificate).
- The cryptographic material can be provided through a JSON Web Key (JWK).
- If a JWK header parameter is included in the header of the JWT, the parameter must match the JWK or cryptographic object that is specified in the policy, or the JWT validation will fail.
- If both a cryptographic object and a JWK are specified, the cryptographic object is used to decrypt or verify the JWT.
- The JWT validate action on the DataPower API Gateway can verify a JWT by using either a single JWK, or a JWK set.

## About

The jwt-validate policy has the following structure:

```
- jwt-validate:
 version: version
 title: title
 description: description
 jwt: json_web_token
 output-claims: output_full_set_of_jwt_claims
 iss-claim: issuer_claim
 aud-claim: audience_claim
 jwe-crypto: decrypt_crypto_object
 jwe-jwk: decrypt_crypto_jwk_variable_name
 jws-crypto: verify_crypto_object
 jws-jwk: verify_crypto_jwk_variable_name
```

## Properties

The following table describes the policy properties:

Table 2. Validate JWT policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	Yes	The title of the policy.	string
description	No	A description of the policy.	string
jwt	Yes	Context or runtime variable that contains the JWT to be validated. The default value is: <code>request.headers.authorization</code> . However, if this property is not set, the policy looks for the JWT in the <code>request.headers.authorization</code> location by default.  Note: The format of the authorization header must be:  <code>"Authorization: Bearer <i>jwt-token</i>"</code>  where <i>jwt-token</i> is the encoded JWT.	string
output-claims	Yes	Runtime variable to which the full set of claims that are contained in the JWT is assigned. The default value is: <code>decoded.claims</code> .	string
iss-claim	No	The Perl Compatible Regular Expressions (PCRE) to use to validate the Issuer (iss) claim.	string
aud-claim	No	The PCRE to use to validate the Audience (aud) claim.	string
jwe-crypto	No	The cryptographic object (a shared key or certificate) to use to decode the claim. A JWK and a Crypto Object are both valid ways of providing the cryptographic data necessary to decrypt the JWT. However, if both data types are specified, only the Crypto Object is used.	string
jwe-jwk	No	Runtime variable that contains the JWK to use to decrypt the JWT. A JWK and a Crypto Object are both valid ways of providing the cryptographic data necessary to decrypt the JWT. However, if both data types are specified, only the Crypto Object is used.	string
jws-crypto	No	The cryptographic object (a shared key or certificate) to use to verify the signature. A JWK, or a JWK set, and a Crypto Object are both valid ways of providing the cryptographic data necessary to verify the JWT. However, if both data types are specified, only the Crypto Object is used.	string
jws-jwk	No	Runtime variable that contains the JWK, or a JWK set, to use to verify the signature. A JWK, or a JWK set, and a Crypto Object are both valid ways of providing the cryptographic data necessary to verify the JWT. However, if both data types are specified, only the Crypto Object is used.	string

## Example

The following is an example of a `jwt-validate` policy:

```
- jwt-validate:
 version: 1.0.0
 title: jwt-validate
 jwt: request.headers.authorization
 output-claims: decoded.claims
 iss-claim: "'^data.*'"
 aud-claim: "'^id.*'"
 jwe-crypto: jweCryptoObjectName
 jwe-jwk: jwe.jwk
 jws-crypto: jwsCryptoObjectName
 jws-jwk: jws.jwk
```

For more information about how to use a `jwt-validate` security policy, see [Validate JWT](#) in the built-in policies section.



## log

Use the `log` policy to customize or override the default activity logging configuration for an API.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0
	2.1.0 (DataPower API Gateway Version 10.0.3.0 or later)

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Log](#).

## About

The `log` policy has the following format:

```
- log:
 version: version
 title: title
 description: description
 mode: activity_logging_actions
 log-level: type_of_content_to_log
```

## Properties

The following table describes the properties of the log policy.

Table 2. log policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
mode	Yes	<p>Specify one of the following values:</p> <ul style="list-style-type: none"><li><b>gather-only</b>: gather all analytics data and write it to the <code>log</code> context variable, which populates the API event record on completion of the API execution. For more information on the fields in the <code>log</code> context variable, and the consequent API event record, see <a href="#">API event record fields</a>.</li><li><b>send-only</b>: perform the following actions:<ul style="list-style-type: none"><li>Read the data from the <code>log</code> context variable.</li><li>Truncate all message payloads and convert to a textual representation.</li><li>Send the data to the analytics server.</li></ul></li><li><b>gather-and-send</b>: perform a <b>gather-only</b> operation, immediately followed by a <b>send-only</b> operation.</li></ul> <p>If you use the Send-only or Gather-and-send option, data is buffered and sent to the analytics server in batches according to the time interval configured for the Analytics Endpoint on the DataPower API Gateway. For more information, see <a href="#">Configuring an analytics endpoint</a> in the DataPower knowledge center.</p> <p>Note: If you are offloading to a third party analytics server, you can redact any aspect of the event data. If you are using API Connect analytics, you can redact only request and response payloads.</p>	string
log-level	No	<p>The type of content to log. Specify one of the following values:</p> <ul style="list-style-type: none"><li><b>none</b>: Indicates no logging.</li><li><b>activity</b>: Logs the invocation only, which is only the resource URI.</li><li><b>header</b>: Logs activity and header.</li><li><b>payload</b>: Logs activity, header, and payload.</li><li><b>default</b>: Use the log level setting defined in the API definition; for more information, see <a href="#">activity-log</a>. This is the default value.</li><li><b>\$(value)</b>: An inline parameter in the format <code>\$(value)</code> to retrieve a value from the API context.</li></ul>	string

## Example

```
- log:
 version: 2.0.0
 title: Gather activity log data for processing
 mode: gather-only
```

## map

Use the map policy to transform your assembly flow and specify relationships between variables.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.5.0.0 or later)

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Map](#).

## About

For information about the use and structure of the map policy, see [The Map policy structure](#). For more information about API properties that affect the map policy, see [API properties](#).

The map policy has the following format:

```
- map
 version: version
 title: title
 description: description

 inputs:
 - input_1:
 variable: context_1
 $ref: '#/definitions/definition_1'
 - input_2
 variable: context_2
```

```

 type: type_2
 content: content_type

outputs:
 - output_3
 variable: context_3
 type: type_3

actions:
 - set: output_3.output_property_3
 from: input_1.input_property_1

 - set: output3.output_property_3
 from:
 - input1.input_property_1
 - input2.input_property_2
 value: 'script_A'
 default: 'default_A'

 - create: output3.output_property_3
 from: input1.input_property_1
 foreach: input1.input_property_1
 actions:
 - further_actions
options:
 .
 .
 .
 advanced_XML_and_general_configuration_options
 .
 .

```

## Properties

Table 2.

Property	Required	Description	Data type	Belongs to
version	Yes	The policy version number	string	N/A
title	No	A title for the policy.	String	N/A
description	No	A policy description.	String	N/A
inputs	No	An array listing the inputs of the map policy.	Object	N/A
outputs	Yes	An array listing the outputs of the map policy.	Object	N/A
variable	Yes	A reference to the context variable that is the location of the input or output variable..	String	inputs or outputs
\$ref	Yes <sup>1</sup>	A reference to the definition of the type of the variable.	String	inputs or outputs
type	Yes <sup>1</sup>	The type of the variable.	String	inputs or outputs
content	No	The content type of the variable: <code>application/xml</code> or <code>application/json</code> . If <code>None</code> is selected, or the field is not included, then the <code>type</code> is treated as JSON.	String	inputs or outputs
actions	Yes	Lists actions to be performed by the map policy.	Object	N/A
set	Yes <sup>2</sup>	Specifies by name the output variable that is to be set to a value by the action.	String	actions
create	Yes <sup>2</sup>	Specifies by name the output variable that is to have a value appended to it by the action.	String	actions
from	No	Specifies by name any input variables used by the action.	String	actions
value	No	Contains a script that maps and transforms input variables into output variables.	String	actions
default	No	Contains a static value, or an inline variable reference, to be applied to the output when no input value is provided. For information on inline variable references, see <a href="#">Inline references</a> .	String	actions
foreach	No	Specifies by name an array for which further actions should be performed for each element.	String	actions
includeEmptyXMLElements	No	If set to <code>true</code> , empty XML elements are included in the output of the map policy. Set to <code>false</code> if you do not want empty XML elements to be included in the output of the map policy. The default value is <code>true</code> .	Boolean	options
namespaceInheritance	No	If set to <code>true</code> , XML namespaces are inherited from the parent element. Set to <code>false</code> if you want the map policy to use explicit namespaces. The default value is <code>true</code> .	Boolean	options
inlineNamespaces	No	If set to <code>true</code> , XML namespaces will be inserted into the document where they are first used. Set to <code>false</code> if you want namespaces to all be defined on the root element. The default value is <code>true</code> .	Boolean	options
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> mapResolveXMLInputDataType	No	If set to <code>true</code> , XML elements whose schema is configured as type boolean or numeric will be converted to that data type. Set to <code>false</code> if you want all XML element values to be returned as a string.	Boolean	options
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> mapParseXMLOutput	No	If set to <code>true</code> , the map policy will write XML output to <code>message.body</code> as a parsed XML document. By default, the XML will be output as an XML string. XML output to any other variable will always be written as an XML string. The default value is <code>false</code> .	Boolean	options

Property	Required	Description	Data type	Belongs to
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> mapXMLEmptyElement	No	<p>This property controls how the map policy handles the output of an empty XML element. Specify one of the following values:</p> <ul style="list-style-type: none"> <li><b>string</b> (the default option): An empty XML element is handled as an empty string.</li> <li><b>null</b>: An empty XML element is handled as a null value.</li> <li><b>none</b>: The data is ignored.</li> <li><div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> <b>string-badgerfish</b>: The value for an empty XML element is considered to be an empty string. The empty string value will be placed into a JSON badgerfish value property.</li> <li><div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> <b>null-badgerfish</b>: The value for an empty XML element is considered to be null. The null value will be placed into a JSON badgerfish value property. A mapping of this element to a JSON output property does not occur unless the <b>mapNullValue</b> property is set to <b>true</b>.</li> </ul>	String	options
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> mapArrayFirstElementValue	No	<p>If set to <b>true</b>, then if an array is encountered in the traversal of the input, only the first element is used. Set to <b>false</b> if you want the map policy to use all array elements. The default value is <b>false</b>.</p>	Boolean	options
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> mapResolveApicVariables	No	<p>If set to <b>true</b>, API Connect variable references found in the map properties are resolved. Set to <b>false</b> if you want the map policy to ignore API Connect variable references in the map policies. The default value is <b>false</b>.</p>	Boolean	options
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> mapNullValue	No	<p>If set to <b>true</b>, an input property value with a null value is mapped to the output document. Set to <b>false</b> if you want the map policy to ignore null input values. The default value is <b>false</b>.</p>	Boolean	options
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> mapOptimizeSchemaDefinition	No	<p>If set to <b>true</b>, complex schema types evaluation handles circular type references in an optimized manner. Set to <b>false</b> to evaluate these schema types in a standard manner. The default value is <b>false</b>.</p>	Boolean	options
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> mapEmulateV4DefaultRequiredProps	No	<p>If set to <b>true</b>, default values are generated in the output for required properties that are either not mapped, or for which there is no input data present, in the following specific cases:</p> <ul style="list-style-type: none"> <li>An array consists of objects that contain one or more required properties.</li> <li>An object which is optional has one or more child properties that are required.</li> </ul> <p>By default, these required properties are not present in the output. If you set this property to <b>true</b>, these required properties will be present in the output. If the output schema defines a <b>default</b> property for the output property then the specified default value is used, otherwise default value is assigned dependent on the data type, as follows:</p> <ul style="list-style-type: none"> <li>String: empty string ("")</li> <li>Number: 0</li> <li>Boolean: false</li> <li>Object: empty object</li> <li>Array: empty array</li> </ul> <p>Example 1</p> <p>The input data has the following array of objects:</p> <pre>[{"a": "value1"}, {"a": "value2", "b": "value3"}]</pre> <p>The output schema defines the output object as having two properties, <b>a</b> and <b>b</b>, of which <b>b</b> is required. The map policy defines the following mappings:</p> <ul style="list-style-type: none"> <li><b>input.array.a</b> to <b>output.array.a</b></li> <li><b>input.array.b</b> to <b>output.array.b</b></li> </ul> <p>If this property is set to <b>true</b>, and <b>b</b> is either not mapped or has no input data present, then <b>b</b> is assigned a default value of an empty string, and the output is as follows:</p> <pre>[{"a": "value1", "b": ""}, {"a": "value2", "b": "value3"}]</pre> <p>Example 2</p> <p>The output schema defines the following structure:</p> <pre>{"a" : {"b" : {"c" : "value1", "d" : "value2"} } }</pre> <p>Property <b>b</b> is optional but property <b>d</b> within <b>b</b> is required.</p> <p>The map policy defines a mapping to <b>output.a.b.c</b>.</p> <p>If this property is set to <b>true</b>, and <b>d</b> is not mapped, then <b>d</b> is assigned a default value of an empty string, and the output is as follows:</p> <pre>{"a" : {"b" : {"c" : "value1", "d" : ""} } }</pre> <p>If this property is set to <b>false</b>, these required properties are <b>not</b> created in the output with their default values.</p> <p>The default value is <b>false</b>.</p>	Boolean	options

Property	Required	Description	Data type	Belongs to
API Gateway only mapEnablePostProcessingJSON	No	If set to <b>true</b> , mapped JSON output is post processed. The post processing of JSON output will use the output schema to ensure that property values are of the same data type as that defined in the schema. It will also normalize output property values that have a Badgerfish JSON syntax due to object mapping of an XML input. Set to <b>false</b> if you want no post processing of mapped JSON output. The default value is <b>false</b> .	Boolean	options
API Gateway only mapCreateEmptyArray	No	This property controls how the map policy handles the output of an empty array. Specify one of the following values: <ul style="list-style-type: none"> <li><b>all</b>: Output all empty arrays, including empty children arrays.</li> <li><b>parent</b>: Output only the current property's empty array value. Children map actions of this property are not attempted.</li> <li><b>none</b>: Prevent any empty output array values from being produced.</li> </ul> The default value is <b>all</b> .	String	options
API Gateway only mapReferenceLimit	No	Set the value of this property to an integer value that specifies the maximum allowed number of iterations of a circular schema definition. The default value is 1, which means that circular schema definitions are not followed. The maximum possible value is 5. If you specify a value greater than 5, a value of 5 is assumed. If you specify a non-numeric value, a value of 1 is assumed.	String	options
messagesInputData	No	This property defines the severity level for log messages that relate to input data. Specify one of the following values: <ul style="list-style-type: none"> <li><b>error</b></li> <li><b>warn</b></li> <li><b>info</b></li> </ul>	String	options
API Gateway only mapEmulateV4EmptyJSONObject	No	If a mapping fails because its input is not present and there is no default mapping configured, the default behavior is to not to make any change to the output mapping. Set this property to <b>true</b> to create an empty object for the parent of the target mapping, emulating the behavior of IBM® API Management Version 4.0.  Example The map policy defines a mapping to <b>output.a.b.c</b> . If input data is present, the output is as follows: <pre> {   "a": {     "b": {       "c": "inputvalue"     }   } } </pre> If there is no input data, and the mapEmulateV4EmptyJSONObject option is set to <b>true</b> , the output is as follows: <pre> {   "a": {     "b": {     }   } } </pre> Properties <b>a</b> and <b>b</b> are created but the value of <b>b</b> is an empty object. The default value is <b>false</b> .	Boolean	options

<sup>1</sup> There must be one of \$ref or type in the description of a variable.

<sup>2</sup> There must be one of set or create in an actions field.

## Example

```

- map:
 version: 1.0.0
 title: Output mapping
 inputs:
 Monthly_cost:
 schema:
 type: double
 variable: loan_invoke.body.monthly_payment
 content: application/json
 Duration:
 schema:
 type: integer
 variable: request.parameters.duration
 outputs:
 Quote_Output:
 schema:
 $ref: '#/definitions/Quote_Output'
 variable: message.body
 content: application/json
 actions:
 - set: Quote_Output.monthly_repayment
 from: Monthly_cost

```

```

value: ''
- set: Quote_Output.total_cost
 from:
 - Duration
 - Monthly_cost
 value: '$(Duration)*$(Monthly_cost)'
description: Maps and transforms contexts to the operation output.
options:
 includeEmptyXMLElements: false
 namespaceInheritance: false
 inlineNamespaces: false
 API Gateway only mapResolveXMLInputDataType: true
 API Gateway only mapXMLEmptyElement: null
 API Gateway only mapArrayFirstElementValue: false
 API Gateway only mapResolveApicVariables: false
 API Gateway only mapNullValue: true
 API Gateway only mapOptimizeSchemaDefinition: true
 API Gateway only mapCreateEmptyArray: parent
 API Gateway only mapReferenceLimit: 5
messagesInputData: warn
mapEmulateV4EmptyJSONObject: true

```

## operation-switch

Use the operation-switch construct when you want to execute alternative policy assemblies, conditional on the operation that is being called.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [operation-switch](#).

## About

An operation can be described with a **verb/path** pair, or with an **operationId**. The **operationIds** are strings, or names, that are defined in the OpenAPI document.

The operation-switch policy has the following format:

```

- operation-switch:
 version: version
 title: title
 description: description
 case:
 - operations:
 - verb: operation_verb_1_1
 path: operation_path_1_1
 - verb: operation_verb_1_2
 path: operation_path_1_2
 .
 .
 further verb/path combinations
 .
 .
 execute:
 policy_assembly_1 ...
 - operations:
 - verb: operation_verb_2_1
 path: operation_path_2_1
 - verb: operation_verb_2_2
 path: operation_path_2_2
 .
 .
 further verb/path combinations
 .
 .
 execute:
 policy_assembly_2 ...
 .
 .
 - operations:
 - operationID_3_1

```



```

- operationID_3_2
- operationID_3_3
.
.
.
 further operationIDs
.
.
.
execute:
 policy_assembly_3 ...
.
.
.
 further operations sections
.
.
.

```

For each `operations:` section, if the operation that is being called matches any of the `verb/path` combinations or `operationId` strings listed in that `operations:` section, then the policy assembly that is defined in the `execute:` section is executed. Therefore, each `operations:` section defines execution of a policy assembly conditional on the operation that is being called.

You can have as many `operations:` sections as you want, and each `operations:` section can have one or more `verb/path` combinations or `operationId` strings.

The `execute:` section can define any policy assembly, including further operation-switch policies.

## Properties

Table 2. operation-switch policy properties

Property	Required	Description	Data type	Contained in
version	Yes	The policy version number	string	N/A
title	No	A title for the policy.	string	N/A
description	No	A policy description.	string	N/A
case	Yes	An array containing different cases, each entry contains an operations and execute field.	array (object)	N/A
operations	Yes	The operations to which a case applies.	object	case
verb	No	An operation verb. Valid values: <ul style="list-style-type: none"> <li>• GET</li> <li>• POST</li> <li>• PUT</li> <li>• DELETE</li> <li>• HEAD</li> <li>• PATCH</li> <li>• OPTIONS</li> </ul>	string	operations
path	No	A relative path to an individual endpoint. For example, <code>/account_status</code> .	string	operations
operationID	No	The operation is defined in OpenAPI. The policy operation refers to the OpenAPI operation.	string	operations
execute	Yes	The policy assembly that you want to execute if the operation that is being called matches any one of the <code>verb/path</code> combinations. For more information, see <a href="#">execute</a> .	string	case

## Example

Example that defines match operations with `verb/path` combinations:

**# carry out different redaction actions depending on the operation**

```

- operation-switch:
 version: 1.0.0
 title: clear_region_and_set_body
 case:
 - operations:
 - verb: GET
 path: /account_details
 execute:
 - redact:
 title: remove secret field
 actions:
 - action: remove
 from: all
 path: /document/user/secret
 - operations:
 - verb: GET
 path: /account_status
 execute:
 - redact:
 title: redact address
 actions:
 - action: redact
 from: response
 path: /*[@name='secondaryAddress']/*[@name='streetAddress']

```

Example that defines match operation with `operationIDs`:

```
match on operationIDs
- operation-switch:
 title: customer_actions
 case:
 - operations:
 - getCustomerByName
 - deleteCustomer
 - addACustomer
 execute:
 .
 .
 .
```

## oauth

Use the oauth policy to policy to perform OAuth processing based on defined OAuth provider settings.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [OAuth](#).

## About

The oauth policy has the following format:

```
- oauth:
 version: version
 title: title
 description: description
 oauth-provider-settings-ref:
 .
 .
 .
 references_to_oauth_settings
 .
 .
 supported-oauth-components:
 - oauth_component_1
 - oauth_component_2
 .
 .
```

Note: You add an oauth policy to the OpenAPI source in a native OAuth provider. For more information, see the following topics:

- [Editing the native OAuth provider configuration using the API Editor](#) (Cloud Manager UI)
- [Editing the native OAuth provider configuration using the API Editor](#) (API Manager UI)

## Properties

Table 2. oauth policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A description of the policy.	string
oauth-provider-settings-ref: default	Yes	The name of an existing OAuth provider that defines the required settings.	string
oauth-provider-settings-ref: url	No	A URL to a document that contains serialized XML or JSON properties that defines OAuth token generate settings. URL reference takes precedence over any existing literal configuration or object reference.	string
oauth-provider-settings-ref: literal	No	A literal string that contains serialized XML or JSON properties that defines OAuth token generate settings. Literal configuration takes precedence over any existing object reference.	string

Property	Required	Description	Data type
supported-oauth-components:	Yes	Specify the OAuth components that are supported by this policy, as follows:	string
- oauth_component_1		- OAuthValidateRequest Validates the authorization request from the client.	
- oauth_component_2		- OAuthGenerateAZCode Generates the authorization code for the client, which represents the resource owner's authorization that grants access to the requested resource.	
.		- OAuthVerifyAZCode Verifies the authorization code from the client.	
.		- OAuthVerifyRefreshToken Verifies the refresh token that is presented by the client.	
.		- OAuthCollectMetadata Collects metadata about a client for later user interaction	
.		- OAuthGenerateAccessToken Generates the access token to the client when the authorization code or refresh token is verified.	
.		- OAuthIntrospectToken Introspects the token to determine its state and, when active, its metadata.	

## Overriding default OAuth provider settings

You can use either the `literal` property or the `url` property to dynamically override any OAuth provider configuration settings to dynamically override any OAuth provider configuration settings defined by the `default` property.

For example, to override the access token expiration time with a value of 200 seconds, include the following configuration in either the literal string or the document at the specified URL:

```
<OAuthProviderSettings><APICAccessTokenTTL>200</APICAccessTokenTTL></OAuthProviderSettings>
```

For a list of all OAuth provider settings, refer to the `OAuthProviderSettings` management schema, defined in the `xml-mgmt.xsd` file located in the store: directory on the DataPower API Gateway.

If you are using the API Manager user interface, the connection details are determined by the API Manager URL that you open, and the user ID with which you log in. If you are using the API Designer user interface, you provide the management server details and user ID in the login window that opens when you first launch API Designer; see [Logging into API Connect Designer](#).

## oauth policy example

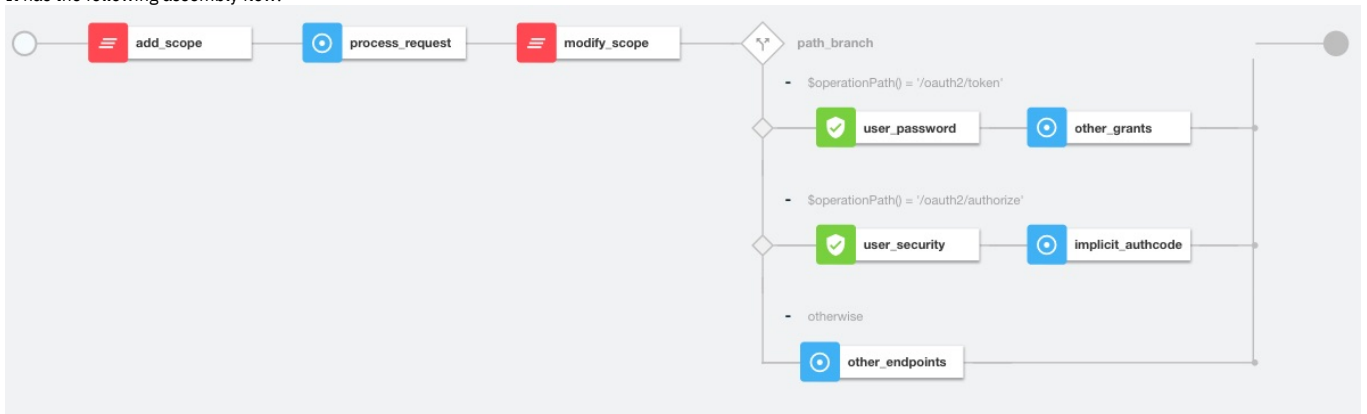
```
- oauth:
 version: 2.0.0
 title: my-oauth-policy
 oauth-provider-settings-ref:
 default: my-oauth
 supported-oauth-components:
 - OAuthGenerateAZCode
 - OAuthGenerateAccessToken
 - OAuthIntrospectToken
 - OAuthVerifyAZCode
 - OAuthVerifyRefreshToken
```

## Example - using multiple OAuth policies in an OAuth provider assembly

This example demonstrates the use of multiple OAuth policies in the assembly flow for a native OAuth provider.

The example is based on the default assembly that is generated when you create a native OAuth provider, and is customized with the addition of `gatewayscript` policies that use [OAuth context variables](#) to manipulate the OAuth flow. For details on creating a native OAuth provider, see [Configuring a native OAuth provider](#).

It has the following assembly flow:



The following sections describe the OpenAPI source code that underlies each of the policies in the assembly; for the complete assembly code, download [multiple\\_oauth\\_policies.txt](#).

## Sample policy to add a custom scope

---

The `add_scope` policy is a `gatewayscript` policy that adds a custom scope to the request.

The underlying OpenAPI source YAML is as follows:

```
- gatewayscript:
 version: 2.0.0
 title: add_scope
 source: |-
 // Add another custom scope to the request
 let scope = context.get("request.parameters.scope.values[0]");
 if (scope)
 context.set("oauth.processing.scope", scope + " custom");
```

## Validate the initial OAuth request

---

The first `process_request` policy is an `oauth` policy that processes the initial request and verifies that the request is valid. The result of the processing is stored automatically in the `oauth.processing` context variables for use, as required, by the next OAuth policy in the assembly flow.

The underlying OpenAPI source YAML is as follows:

```
- oauth:
 title: process_request
 version: 2.0.0
 description: >-
 This oauth policy performs all OAuth/OpenID Connect protocol steps
 that are needed for OAuth Validation by default. The inputs and
 outputs of each of the steps are driven by documented context
 variables. Add or remove the Supported OAuth Components as required.
 oauth-provider-settings-ref:
 default: custom-form
 supported-oauth-components:
 - OAuthValidateRequest
```

## Sample policy to modify the scope

---

The `modify_scope` policy is a `gatewayscript` policy that modifies the scope depending on the calling application.

The underlying OpenAPI source YAML is as follows:

```
- gatewayscript:
 version: 2.0.0
 title: modify_scope
 source: |-
 let admin_id = '1f1a2aa4-db9f-4423-b2f1-e2572b12123a';

 // Check application and modify the scope
 let app = context.get("oauth.processing.client_id");
 let scope = context.get("oauth.processing.scope");
 if (app === admin_id) {
 context.set("oauth.processing.scope", scope + " admin");
 } else {
 context.set("oauth.processing.scope", scope + " customer");
 }
 }
```

## Branch conditionally according to the OAuth path

---

The `path_branch` policy is a `switch` policy that branches according to the different OAuth paths to process the resource owner.

The underlying OpenAPI source YAML is as follows:

```
- switch:
 version: 2.0.0
 title: path_branch
 case:
 - condition: ($operationPath() = '/oauth2/token')
 execute:
 .
 .
 .
 definition of the user_security and other_grants policies
 .
 .
 - condition: ($operationPath() = '/oauth2/authorize')
 execute:
 .
 .
 .
 definition of the user_password and implicit_authcode policies
 .
 .
 - otherwise:
 .
```

```
.
.
definition of the other_endpoints policy
.
.
```

## Process the user name and password, and enable grant type component

---

The following two policies operate on the token endpoint.

- The `user_password` policy for password grant type processes the user name and password from the `x-www-form-urlencoded` body. The authentication method is derived from the User Security settings in the OAuth provider; see [Configuring user security for a native OAuth provider](#).
- The `other_grants` policy is an `oauth` policy that enables the `OAuthGenerateAccessToken`, `OAuthVerifyAZCode`, `OAuthVerifyRefreshToken`, and `OAuthCollectMetadata` components to perform the operations for client credentials, authorization code, refresh token, and password grant types.

The underlying OpenAPI source YAML is as follows:

```
- user-security:
 title: user_password
 version: 2.0.0
 description: ''
 factor-id: default
 extract-identity-method: context-var
 user-context-var: request.parameters.username.values
 pass-context-var: request.parameters.password.values
 ei-stop-on-error: false
 user-auth-method: auth-url
 au-stop-on-error: false
 auth-url: 'http://httpbin.org/basic-auth/user/pass'
 user-az-method: authenticated
 az-stop-on-error: true
 auth-response-headers-pattern: (?)*x-api*
 auth-response-header-credential: X-API-Authenticated-Credential
- oauth:
 title: other_grants
 version: 2.0.0
 description: >-
 This oauth policy performs all OAuth/OpenID Connect
 protocol steps that are needed for token path by default.
 The inputs and outputs of each of the steps are driven by
 documented context variables. Add or remove the Supported
 OAuth Components as required.
 oauth-provider-settings-ref:
 default: custom-form
 supported-oauth-components:
 - OAuthGenerateAccessToken
 - OAuthVerifyAZCode
 - OAuthVerifyRefreshToken
 - OAuthCollectMetadata
```

## Perform authorization checks, and enable grant type components

---

These following two policies operate on the authorize endpoint.

- The `user_security` policy configuration is derived from the User Security settings in the OAuth provider; see [Configuring user security for a native OAuth provider](#).
- The `implicit_authcode` policy is an `oauth` policy that enables the `OAuthGenerateAZCode`, `OAuthGenerateAccessToken`, and `OAuthCollectMetadata` components to perform the operations for the implicit and authorization code grant types.

The underlying OpenAPI source YAML is as follows:

```
- user-security:
 title: user_security
 version: 2.0.0
 description: >-
 This user security policy performs EI(basic) and AU(auth
 url) check for oauth assembly. Change the security check
 method as required
 factor-id: default
 extract-identity-method: basic
 ei-stop-on-error: true
 user-auth-method: auth-url
 au-stop-on-error: true
 user-az-method: authenticated
 az-stop-on-error: true
 auth-response-headers-pattern: (?)*x-api*
 auth-response-header-credential: X-API-Authenticated-Credential
 auth-url: 'http://httpbin.org/basic-auth/user/pass'
- oauth:
 title: implicit_authcode
 version: 2.0.0
 description: >-
 This oauth policy performs all OAuth/OpenID Connect
 protocol steps that are needed for az code path by
 default. The inputs and outputs of each of the steps are
 driven by documented context variables. Add or remove the
 Supported OAuth Components as required.
 oauth-provider-settings-ref:
 default: custom-form
 supported-oauth-components:
```

- OAuthGenerateAZCode
- OAuthGenerateAccessToken
- OAuthCollectMetadata

## Process all other endpoints

The `otherwise` condition catches all other endpoints such as the introspect and revoke endpoints. The `other_endpoints` policy in the `otherwise` condition is an `oauth` policy that enables the `OAuthIntrospectToken`, and `OAuthRevokeTokencomponents` components to perform the operations for introspect and revoke.

The underlying OpenAPI source YAML is as follows:

```
- oauth:
 title: other_endpoints
 version: 2.0.0
 description: >-
 This oauth policy performs all OAuth/OpenID Connect
 protocol steps that are needed for all other paths by
 default. The inputs and outputs of each of the steps are
 driven by documented context variables. Add or remove the
 Supported OAuth Components as required.
 oauth-provider-settings-ref:
 default: custom-form
 supported-oauth-components:
 - OAuthIntrospectToken
 - OAuthRevokeToken
```

## Related concepts

- [API policies and logic constructs](#)

## Related reference

- [OAuth context variables](#)

## Related information

- [Configuring a native OAuth provider](#)

## parse

Use the parse policy to control the parsing of an input document. When the input document is a JSON string, the string is parsed instead of copied over.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0
	2.1.0 (DataPower API Gateway Version 10.0.3.0 or later)
	2.2.0 (DataPower API Gateway Version 10.5.0.5 or later)

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Parse](#).

## About parse

The parse policy has the following format:

```
- parse:
 version: version
 title: title
 description: description
 use-content-type: request_header_usage_setting
 parse-settings-reference:
 .
 .
 .
 references_to_parse_settings
 .
 .
 .
 input: input_message
 output: output_message
```

## Properties

Table 2. parse policy properties

Property	Required	Description	Data type
version	Yes	The policy version number.	string
title	No	The title of the policy.	string

Property	Required	Description	Data type
description	No	A description of the policy.	string
use-content-type	No	<p>If this setting is enabled and the parse setting is configured to detect the document type, the parse operation uses the <b>Content-Type</b> specified in the request headers.</p> <p>If this setting is enabled and the document type in the parse setting is configured for either JSON or XML, the parse operation uses the <b>Content-Type</b> specified in the request headers and fails if the <b>Content-Type</b> in the request headers does not match the parse setting.</p> <p>Enabling this setting is applicable only when the expected <b>Content-Type</b> is either JSON or XML.</p> <p>If this setting is not enabled, the parse operation uses either the document type specified in the parse setting, or the detected document type if the parse setting is configured to detect the document type.</p> <p>The default value is <b>false</b>.</p>	boolean
parse-settings-reference-default	No	An existing valid object from which to retrieve default property values for the dynamic object.	string
document_type	No	<p>The type of document to parse. Specify one of the following values:</p> <ul style="list-style-type: none"> <li><b>detect</b>: the document type is detected and the payload is parsed accordingly. This type is the default option.</li> <li><b>JSON</b>: the payload is parsed as JSON.</li> <li><b>XML</b>: the payload is parsed as XML.</li> <li><b>graphql</b>: parsing is performed in one of the following ways: <ul style="list-style-type: none"> <li>As GraphQL in the body of a POST request. The GraphQL is in the following format: <pre>{   me {     name   } }</pre> </li> <li>As a JSON string in the body of a POST request that contains a GraphQL query. The JSON object is in the following format: <pre>{   "query": "query MyData (\$idVar: Int!) { user (id: \$idVar) { name }}",   "operationName": "MyData",   "variables": {"idVar": 123} }</pre> </li> <li>Through the URL query of a GET request. The URL is in the following format: <pre>https://myapi/graphql?query={me{name}}</pre> <p>The following example includes the <b>variables</b> and <b>operationName</b> properties, with URL encoding:</p> <pre>https://hostname/basepath/graphql? query=query%20fetchAccounts%20(\$limit:%20Int)%20(accounts(limit:%20\$limit) %20{name{first,last}})&amp;variables={"limit":100}&amp;operationName=fetchAccounts</pre> </li> </ul> </li> </ul> <p>Query variables in a GET request can be sent as a JSON string in a query parameter called <b>variables</b>. If the query contains several named operations, an <b>operationName</b> query parameter is required to determine which operation is to be executed.</p>	string
max_doc_size	No	<p>Optionally, the maximum document size in bytes that the parse operation accepts. This setting provides threat protection by enforcing the maximum document size. Enter a value in the range 0 - 5368709121. A document is rejected when its size exceeds the maximum size. A value of 0 indicates an unlimited document size. When set to 0, the operation result does not return the document size. This setting is applicable to JSON, XML, or GraphQL input documents.</p> <p>The default value is 4194304.</p>	integer
max_nesting_depth	No	<p>Optionally, the maximum nesting depth of an XML, JSON, or GraphQL message that the parse operation accepts. This setting provides threat protection by enforcing limits in the message.</p> <ul style="list-style-type: none"> <li>When XML, it is the maximum level of element depth.</li> <li>When JSON, it is the maximum level of nested label-value pairs, the maximum number of nested arrays, or the maximum number of combination of label-value pairs and arrays.</li> <li>When GraphQL, it is the maximum level of nested selection sets.</li> </ul> <p>Enter a value in the range 0 - 4096. A document is rejected when its nesting depth exceeds the maximum depth. A value of 0 indicates unlimited nesting depth. When set to 0, the operation result does not return the nesting depth.</p> <p>The default value is 512.</p>	integer

Property	Required	Description	Data type
max_width	No	<p>Optionally, the maximum width of the payload that the parse operation accepts.</p> <ul style="list-style-type: none"> <li>When the input document is XML, this property specifies the maximum number of attributes for an element or the maximum number of child elements for an element.</li> <li>When the input document is JSON, this property specifies the maximum number of properties on a JSON object or the maximum number of JSON items in a JSON array.</li> <li>When the input document is GraphQL, this property specifies the maximum number of selections in a selection set.</li> </ul> <p>Enter a value in the range 0 - 65535. The document is rejected when its width exceeds the maximum width. A value of 0 indicates unlimited width. When set to 0, the operation result does not return the width of the document.</p> <p>The default value is 4096.</p>	integer
max_name_length	No	<p>Optionally, the maximum name length in bytes in a document that the parse operation accepts.</p> <ul style="list-style-type: none"> <li>For XML, it is the length of the name portion of a tag.</li> <li>For JSON, it is the length of the label portion of the JSON label-value pair.</li> <li>For GraphQL, it is the maximum length of the identifiers, including field names and directive names.</li> </ul> <p>The length includes any white space that is contained between tags in XML or quotation marks in JSON. Enter a value in the range 0 - 8192. A document is rejected when its name length exceeds the maximum length. A value of 0 indicates unlimited name length. When set to 0, the operation result does not return the name length of a document.</p> <p>The default value is 256.</p>	integer
max_value_length	No	<p>Optionally, the maximum value length in bytes in a document that the parse operation accepts.</p> <ul style="list-style-type: none"> <li>For XML, it is the length of an attribute or the length of a text value.</li> <li>For JSON and GraphQL, it is the length of a string value.</li> </ul> <p>The length includes any white space that is contained between tags in XML or quotation marks in JSON. Enter a value in the range 0 - 5368709121. A document is rejected when its value length exceeds the maximum length. A value of 0 indicates unlimited value length. When set to 0, the operation result does not return the value length of a document.</p> <p>The default value is 8192.</p>	integer
max_unique_names	No	<p>Optionally, the maximum number of unique names in a JSON or XML document that the parse operation accepts.</p> <ul style="list-style-type: none"> <li>For XML, it is the number of unique XML local names.</li> <li>For JSON, it is the number of unique JSON labels.</li> </ul> <p>Enter a value in the range 0 - 1048575. A document is rejected when the number of unique names in the document exceeds the maximum number. A value of 0 indicates an unlimited number of unique names. When set to 0, the operation result does not return the number of unique names in a document.</p> <p>The default value is 1024.</p>	integer
max_unique_prefixes	No	<p>Optionally, the maximum number of unique XML namespace prefixes in a document that the parse operation accepts. This limit counts multiple prefixes defined for the same namespace, but does not count multiple namespaces defined in different parts of the input document under a single prefix. Enter a value in the range 0 - 262143. A document is rejected when the number of unique prefixes in the document exceeds the maximum number. A value of 0 indicates an unlimited number of unique prefixes. When set to 0, the operation result does not return the number of unique prefixes in a document.</p> <p>The default value is 1024.</p>	integer
max_unique_namespaces	No	<p>Optionally, the maximum number of unique XML namespace URIs that the parse operation accepts. This limit counts all XML namespaces, regardless of how many prefixes are used to declare them. Enter a value in the range 0 - 65535. A document is rejected when the number of unique namespaces in the document exceeds the maximum number. A value of 0 indicates an unlimited number of unique namespaces. When set to 0, the operation result does not return the number of unique namespaces in a document.</p> <p>The default value is 1024.</p>	integer
max_number_length	No	<p>Optionally, the maximum number of bytes in the value portion of a JSON label-value pair when the value is a number. The number must be a contiguous string of bytes that contain no white space. The number can include a minus sign and a positive or negative exponent. Enter a value in the range 0 - 256. A document is rejected when the number length in the document exceeds the maximum length. A value of 0 indicates unlimited number length. When set to 0, the operation result does not return the number length in a document.</p> <p>The default value is 128.</p>	integer
strict-utf8-encoding	No	<p>For JSON documents, whether to enforce strict UTF-8 encoding throughout the entire document.</p> <ul style="list-style-type: none"> <li>When <b>on</b>, the entire document is checked for valid UTF-8 encoding.</li> <li>When <b>off</b>, only the first few bytes are checked for proper encoding. The rest of the document is assumed to be in the same encoding.</li> </ul> <p>The default value is <b>off</b>.</p>	boolean
parse-settings-reference: literal	No	<p>A literal string as serialized XML or JSON properties that are merged into the dynamic object. These properties take preference over any existing default properties.</p>	string
parse-settings-reference: url	No	<p>A URL that represents a named context from which to retrieve the serialized XML or JSON properties that are merged into the dynamic object. These properties take preference over any existing literal or default properties.</p>	string



Property	Required	Description	Data type
input	No	The name of a variable in the API context. The content of the <b>body</b> field of the variable is the input to the policy. The default variable name is <b>message</b> .	string
output	No	The name of a variable in the API context. The content of the body field of the variable is the output of the parse operation. The parse metrics of the parsed document can be stored in a different part of the message. The default variable name is the same as the input name, so by default the input message is overwritten by the output message.	string

## parse policy example

```
- parse:
 version: 2.0.0
 title: my-parse-policy
 use-content-type: true
 parse-settings-reference:
 default: my-parse
 input: input-message
 output: output-message
```

## parse-settings-reference: literal example

In XML:

```
- parse:
 version: 2.0.0
 title: parse
 parse-settings-reference:
 default: apic-default-parsesettings
 literal: <ParseSettings><DocumentType>xml</DocumentType></ParseSettings>
```

In JSON:

```
- parse:
 version: 2.0.0
 title: parse
 parse-settings-reference:
 default: apic-default-parsesettings
 literal: { \"ParseSettings\" : { \"propertyName\" : propertyValue } }
```

## proxy

Apply the proxy policy to proxy another API within your operation, particularly if you need to call a large payload.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway, functionality provided by <a href="#">Invoke</a>	

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Proxy](#).

## About

Keep the following considerations in mind regarding the proxy policy.

- Only one proxy policy is permitted to be called per assembly.
- More than one proxy policy can be applied, if they are contained in mutually exclusive branches of the assembly.
- You can use the proxy policy to return multipart form data, that is, when the response is set to `Content-Type: multipart/related`. However, proxy must be the final policy in the assembly, otherwise the response that is received is manipulated causing the multipart form data to be lost.
- The proxy policy, if inside a conditional policy, must be the **final** policy to be executed in the API. If you need further processing afterward, use the [invoke](#) policy rather than the proxy policy.
- The Proxy policy does not currently attempt to rewrite a Location header that is returned from the back end.

The proxy policy has the following structure:

```
- proxy:
 version: version
 title: title
 description: description
 target-url: URL_of_target_API
 tls-profile: TLS_profile_to_be_used
 verb: method_type
 http-version: HTTP_version
 timeout: timeout_value_in_seconds
 compression: is_data_to_be_compressed
 username: username_if_authentication_required
```

```

password: password_if_authentication_required
output: location_of_the_proxy_result
cache-key: unique_identifier_of_the_document_cache_entry
cache-response: cache_behavior
cache-putpost-response: response_caching_behavior
cache-ttl: cache_time_to_live
stop-on-error: errors_that_stop_the_flow

```

## Properties

The table describes the properties of the proxy policy.

Table 2. Proxy policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
target-url	Yes	The URL of the target API.	string
tls-profile	No	The TLS profile to be used.	string
verb	No	The operation method type.  Valid values: <ul style="list-style-type: none"> <li>Keep</li> <li>GET</li> <li>POST</li> <li>PUT</li> <li>DELETE</li> <li>PATCH</li> <li>HEAD</li> <li>OPTIONS</li> </ul> The default value is <b>Keep</b> .	string
http-version	No	The HTTP version. The default value is <b>1.1</b> .	string
timeout	No	The timeout value in seconds. The default value is <b>60</b> .	integer
compression	No	Specifies whether data is to be compressed by using <b>gzip</b> before it is uploaded. The default value is <b>false</b> .	boolean
username	No	The user name, if authentication is required.	string
password	No	The password, if authentication is required.	string
output	No	Specifies the location of the proxy result. By default, the proxy result, that is the body, headers, statusCode and statusMessage, is saved in the variable <b>context.message</b> . The assembly developers can specify an additional location to store the proxy result with the output property.	string
cache-key	No	Specifies the unique identifier of the document cache entry.	string
cache-response	No	The cache response type. Valid values: <ul style="list-style-type: none"> <li>protocol: The cache behavior is defined by the Cache-Control headers on the request and response.</li> <li>no-cache: Specifies that there is no caching. However, if the document is already in the cache, the document is retrieved from the cache.</li> <li>time-to-live: Specifies that the response stays in the cache for the specified time.</li> </ul> The default value is <b>protocol</b> .	string
cache-putpost-response	No	Specifies whether to cache the response from POST and PUT requests. Caching the response from POST and PUT requests can reduce server load and reduce latency in the response to the client request. The default value is <b>false</b> .	boolean
cache-ttl	No	Specifies the amount of time in seconds that the response stays in the cache. Applies only if the property cache-response is set to <b>time-to-live</b> . Enter a value in the range 5 - 31708800. The default value is <b>900</b> .	integer
stop-on-error	No	List the errors that, if thrown during the policy execution, cause the flow to stop. If there is a <b>catch</b> flow configured for the error, it is triggered to handle the error thrown. If an error is thrown and there are no errors specified for the Stop on error property, or if the error thrown is not one of the specified errors, the policy execution is allowed to complete, and the assembly flow continues.	string

API Gateway only

## ratelimit

Use the ratelimit policy to apply one or more rate, burst, or count limits at any point in your API assembly flow. Rate and burst limits restrict the number of calls made to an API in a specified time period, while count limits impose a strict limit on the total number of calls.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
---------	----------------

Gateway	Policy version
DataPower® API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.1.0 or later) 2.2.0 (DataPower API Gateway Version 10.0.2.0 or later) 2.3.0 (DataPower API Gateway Version 10.5.0.0)

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Rate Limit](#).

## About

The defined rate, burst, and count limits are applied to whatever follows in the assembly flow. For example, if a ratelimit policy is placed before an invoke policy, and the call made by the invoke policy exceeds the limits defined by the ratelimit policy, the API call itself fails.

The ratelimit policy has the following format:

```
- ratelimit:
 version: version
 title: title
 description: description
 source: rate_and_burst_limit_location
 rate-limit: rate_limits_to_apply
 burst-limit: burst_limits_to_apply
```

## Properties

The following table describes the properties of the ratelimit policy.

Table 2. ratelimit policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
source	Yes	The location of all the rate limit, burst limit, and count limit definitions that are included in this policy. Specify one of the following values: <ul style="list-style-type: none"> <li><b>catalog-named:</b> the rate or burst limits to be applied are defined in the appropriate <b>api-collection</b> object on the DataPower API Gateway, which is the object that represents your API Connect Catalog in the gateway configuration. For details of how to configure a rate, burst, or count limit in the <b>api-collection</b> object, see <a href="#">Configuring a rate, burst, or count limit on the DataPower API Gateway</a>.</li> <li><b>plan-named:</b> the limits to be applied are assembly burst limits or assembly count limits defined on the Plan to which the calling application is subscribed. For details on how to configure assembly burst limits and assembly count limits in a Plan, see <a href="#">Editing a draft Product</a>. Note: The following conditions apply to a <b>plan-named</b> rate limit policy: <ul style="list-style-type: none"> <li>Every Plan that contains the API that is being limited must have the named rate limit applied.</li> <li>A <b>plan-named</b> rate limit policy cannot be used with automatically generated Products.</li> </ul> </li> <li><b>gateway-named:</b> the limits to be applied are defined in the <b>apigw</b> object, named <b>apiconnect</b>, on the DataPower API Gateway gateway. For details of how to configure a rate, burst, or count limit in the <b>apigw</b> object, see <a href="#">Configuring a rate, burst, or count limit on the DataPower API Gateway</a>.</li> <li><b>plan-default:</b> the rate and burst limits that are applied are the default ones configured in the Plan to which the calling application is subscribed. For details on how to configure default Plan rate and burst limits, see <a href="#">Editing a draft Product</a>. Note: The use of this option does not disable any automatic preflow rate limit policy that might be in place, and could result in the application of the rate limiting being repeated. For more information on preflow policies, see <a href="#">Customizing the preflow policies</a>.</li> </ul>	string
rate-limit	Yes <sup>1</sup>	If the <b>source</b> property is set to <b>catalog-named</b> , a list of rate limit names as defined in the DataPower API Gateway configuration, <sup>1</sup> You must provide at least one rate, burst, or count limit.	string
burst-limit	Yes <sup>2</sup>	If the <b>source</b> property is set to <b>catalog-named</b> , a list of burst limit names as defined in the DataPower API Gateway configuration. If <b>source</b> is set to <b>plan-named</b> , a list of names of assembly burst limits defined in a Plan. <sup>2</sup> You must provide at least one rate, burst, or count limit.	string
count-limit	Yes <sup>3</sup>	If <b>source</b> property is set to <b>catalog-named</b> , a list of count limit names as defined in the DataPower API Gateway configuration. If <b>source</b> is set to <b>plan-named</b> , a list of names of assembly count limits defined in a Plan. <sup>3</sup> You must provide at least one rate, burst, or count limit..	string

Property	Required	Description	Data type
operation	No	<p>For a rate limit, specify one of the following values:</p> <ul style="list-style-type: none"> <li><b>consume</b>: the policy reduces the balance of available requests that remain in the interval defined by the rate limit specified by the <code>rate-limit</code> property. The amount by which the remaining balance is reduced is the value that results from the weight expression defined in the specified rate limit.</li> <li><b>replenish</b>: the policy increases the balance of available requests that remain in the interval defined by the rate limit specified by the <code>rate-limit</code> property. The amount by which the remaining balance is increased is the value that results from the weight expression defined in the specified rate limit.</li> </ul> <p>The purpose of the replenish operation is to restore an appropriate amount of remaining balance in cases where the amount calculated by an earlier consume operation turns out to be too high. For example, for an assembly rate limit with a limit of 100, a consume operation with a weight of 60 uses 60 requests, leaving 40 requests remaining for the interval. If the weight expression in the subsequent replenish operation computes to 15, meaning that the weight expression calculated that the actual request usage was only 45, the replenish operation increases the remaining requests by 15, to a remaining balance of 55.</p> <p>Note that the replenish operation cannot increase the remaining requests allowed beyond the original limit defined in the assembly rate limit scheme. In this example, a replenish operation cannot replenish requests beyond 100. A replenish operation is skipped if the weight expression evaluates to a value of less than 1.</p> <p>The default value is <b>consume</b>.</p> <p>For a count limit, specify one of the following values:</p> <ul style="list-style-type: none"> <li><b>inc</b>: the current count is incremented by the value that results from the weight expression defined in the count limit specified by the <code>count-limit</code> property.</li> <li><b>dec</b>: the current count is decremented by the value that results from the weight expression defined in the count limit specified in the <code>count-limit</code> field.</li> </ul> <p>By default, the count limit is automatically decremented at the end of the API assembly by the total amount incremented during the assembly, minus any decrement operations that are defined in the assembly. You do not need to explicitly define decrement operations unless you want the decrements to occur at specific points in the assembly. However, if the auto decrement feature has been explicitly disabled in the count limit configuration, you must include a corresponding decrement operation, otherwise when the count limit is reached all future calls are blocked. For more information on configuring a count limit, see <a href="#">Configuring a rate, burst, or count limit on the DataPower API Gateway</a>.</p> <p>The default value is <b>inc</b>.</p> <p>For a burst limit, the only possible value is <b>consume</b>, meaning that the limit applied is that defined in the burst limit specified by the <code>burst-limit</code> field.</p>	string

## Example

```
- ratelimit:
 version: 2.0.0
 title: Apply rate and burst limits
 source: catalog-named
 rate-limit:
 - 30perMinute
 - 2000per3Hours
 burst-limit:
 - 5per10Seconds
```



## redact - DataPower API Gateway

Use the redact policy to completely remove or to redact specified fields from the request body, the response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.

## Gateway support

Note: This page describes the redact policy implementation in the DataPower® API Gateway. If you are using the DataPower Gateway (v5 compatible), see [redact - DataPower Gateway \(v5 compatible\)](#).

For information on the different types of gateway, see [API Connect gateway types](#).

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Redaction - DataPower API Gateway](#).

## About

The redact policy has the following format:

```
- redact:
 version: version
 title: title
 description: description
 redactions:
 - action: remove_or_redact
 path: JSONata_expression_for_field_to_remove_or_redact
 .
 .
 .
 root: content_source
```

Note: With the DataPower API Gateway, the input to the redact policy must be parsed data. One way to produce parsed data is to use a [parse](#) policy before a redact policy in your assembly flow, which provides explicit control of the parse action.

## Properties

The following table describes the properties of the redact policy.

Table 2. redact policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
root	No	Specifies the data source that contains the content to which the redact or remove action applies. If the root property is omitted, the action is applied to the entire API context. You can use any supported JSONata path expression.  If you want to apply the action to either request or response data, specify a value of <code>message.body</code> . The actual content to which the action is applied then depends on the positioning of the redact policy in the overall assembly flow; for example: <ul style="list-style-type: none"><li>• If positioned at the beginning, the action is applied to the client request.</li><li>• If positioned after an invoke policy, the action is applied to the response from the back end.</li><li>• If positioned at the end, the action is applied to the response that is returned to the client.</li></ul> If, in your assembly flow, the redact policy is used after a <a href="#">log</a> policy that specifies <code>gather-only</code> for the mode property, specify a root value of <code>log.request_body</code> for the logged request payload, or <code>log.response_body</code> for the logged response payload.	string
path	Yes	Specifies a JSONata path expression that identifies the fields to redact or remove from the source. For more information, see <a href="#">Constructing JSONata expressions to redact fields</a>	string
action	Yes	Specifies whether you want to remove or redact the content. Supply one of the following values: <ul style="list-style-type: none"><li>• <b>remove</b>: Completely removes the specified fields.</li><li>• <b>redact</b>: Redacts (obfuscates with "*"s) the fields to block out the data.</li></ul> The default value is <b>redact</b> .  Note: If a numerical value is being redacted, the redacted value is depicted as <b>*****</b> and the type is changed to <b>string</b> .	string

## Example

```
Specify separate remove and redact actions
```

```
- redact:
 version: 2.0.0
 title: remove price, redact author
 redactions:
 - action: remove
 path: xpath($, '//price')
 - action: redact
 path: $.**.author"
 root: message.body
```

DataPower Gateway (v5 compatible)

## redact - DataPower Gateway (v5 compatible)

Use the redact policy to completely remove or to redact specified fields from the request body, the response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.

## Gateway support

Note: This page describes the redact policy implementation in the DataPower® Gateway (v5 compatible). If you are using the DataPower API Gateway, see [redact - DataPower API Gateway](#).

For information on the different types of gateway, see [API Connect gateway types](#).

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower Gateway (v5 compatible)	1.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Redaction - DataPower Gateway \(v5 compatible\)](#).

## About

The redaction policy has the following format:

```
- redact:
 version: version
 title: title
 description: description
 actions:
 - action: remove_or_redact
 from:
 - where the redaction is to be applied
 path: XPath_expression_for_field_to_remove_or_redact
 .
 .
 .
 further action/from/path combinations
 .
 .
 .
```

You can specify as many `action/from/path` combinations as you want.

## Properties

The following table describes the policy properties:

Table 2. redact policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
action	No	Specifies whether you want to remove or redact the fields. Valid values: <ul style="list-style-type: none"><li><b>remove</b>: Completely removes the specified fields.</li><li><b>redact</b>: Redacts (obfuscates with "*"s) the fields to block out the data.</li></ul> The default value is <b>redact</b> . Note: If a numerical value is being redacted, the redacted value is depicted as <b>*****</b> and the type is changed to <b>string</b> .	string
from	No	Determines where the redaction is to be applied. Valid values: <ul style="list-style-type: none"><li><b>all</b>: Apply the redaction to the request body, the response body, and the activity logs.</li><li><b>request</b>: Apply the redaction to the request body only.</li><li><b>response</b>: Apply the redaction to the response body only.</li><li><b>logs</b>: Apply the redaction to the activity logs only.</li></ul> You can supply one or more values. The default value is <b>all</b> .	string
path	Yes	Specifies an XPath expression that defines the fields to remove or redact. You can construct an XPath expression that is based on JSON or XML depending on whether your API requests and responses use a JSON or an XML format. If the payload is JSON, use the DataPower XML representation of the JSON content (JSONx) to construct the expression. Note: Use a JSONx representation only to identify the XPath expressions for the fields to remove or redact. Do not change the format of any response bodies in API Manager. To learn more about constructing XPath expressions that are based on JSON or XML, see <a href="#">Constructing XPath expressions to redact fields</a> .	string

## Example

```
Specify separate remove and redact actions
```

```
- redact:
 version: 1.0.0
```

```

title: remove secret field, redact address
actions:
- action: remove
 from:
 - all
 path: /document/user/secret
- action: redact
 from:
 - request
 - response
 path: /**[@name='secondaryAddress']/*[@name='streetAddress']

```

## set-variable

Use the set-variable policy to set the value of a runtime variable, or to add or clear a runtime variable.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Set Variable](#).

## About

The set-variable policy has the following format:

```

- set-variable:
 version: version
 title: title
 description: description
 actions:
 - action_type: variable_name
 value: value
 type: data_type

```

## Properties

Table 2. set-variable policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
actions	Yes	Lists the actions to be performed by the set-variable policy.	array
set	Yes <sup>1</sup>	For setting a variable. specifies the name of the variable that you want to set. <sup>1</sup> One of the properties set, add, or clear is required.	string
add	Yes <sup>2</sup>	For adding a variable. specifies the name of the variable that you want to add. <sup>2</sup> One of the properties set, add, or clear is required.	string
clear	Yes <sup>3</sup>	For clearing a variable. specifies the name of the variable that you want to clear. <sup>3</sup> One of the properties set, add, or clear is required.	string
value	Yes <sup>4</sup>	Allocates this value to the specified variable. Can be a literal value, or another variable. <sup>4</sup> value is required only when <b>set</b> or <b>add</b> is specified as the action.	string
<b>API Gateway only</b> type	Yes	Specifies the data type of the variable. Valid values: <ul style="list-style-type: none"> <li><b>any</b></li> <li><b>string</b></li> <li><b>number</b></li> <li><b>boolean</b></li> </ul> For all values other than <b>any</b> , the value is validated against the specified data type. If the data type is specified as <b>boolean</b> , the value property must be set to <b>true</b> or <b>false</b> .	string

## Example 1

```

clear a variable

set-variable:
 version: 1.0.0
 title: clear_region
 actions:
 - clear: message.headers.region

```

## Example 2

```
set a variable to the value of an API Gateway context variable

set-variable:
 version: 2.0.0
 title: set content type
 actions:
 - set: message.headers.contenttype
 value: $(message.headers.content-type)
 type: string
```

## Example 3

```
add a variable

assembly:
 execute:
 - set-variable:
 version: 2.0.0
 title: set-variable
 actions:
 - value: testing add
 add: message.headers.jja
 type: string
```

## switch

Use the switch component to execute one of a number of sections of the assembly based on which specified condition is fulfilled.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [switch](#).

## About



The switch policy has the following format:

```
- switch:
 version: version
 title: switch
 description: 'Description'
 case:
 - condition: Script_1
 execute:
 Assembly_Section_1
 - condition: Script_2
 execute:
 Assembly_Section_2
 - otherwise:
 Assembly_Section_3
```

The **execute** section can define any policy assembly, including further switch policies. For more information, see [execute](#).

## Properties

Table 2. switch policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
case	Yes	Contains the condition and execute pairs of the switch policy.	string
condition	Yes (one or more)	A script that returns <b>true</b> or <b>false</b> .  Use the JSONata expression language to define your condition. See <a href="#">Writing switch condition scripts - DataPower API Gateway</a> .  Use GatewayScript to define your condition. See <a href="#">Writing switch condition scripts - DataPower Gateway (v5 compatible)</a> .	string
execute	Yes (one per condition)	The policy assembly that you want to execute if the condition returns <b>true</b> . For more information, see <a href="#">execute</a> .	string



Property	Required	Description	Data type
otherwise	No	The case you want to execute if no other cases are fulfilled. It functions in the same manner as an execute property. For more information, see <a href="#">execute</a> .	string

For examples, see one or other of the following topics, depending on which gateway type you are using:

- [Writing switch condition scripts - DataPower API Gateway](#)
- [Writing switch condition scripts - DataPower Gateway \(v5 compatible\)](#)



## Writing switch condition scripts - DataPower API Gateway

You write condition scripts for the switch policy in the DataPower® API Gateway by using the JSONata expression language.

The payload of the message must be parsed before you can use JSONata in a switch policy condition. Therefore, a parse policy must precede the switch policy. You can then use a JSONata condition on the parsed content that is stored in the `body` field of the output variable. The default variable is `message`. For example, if the output variable is `message`, you can use the JSONata condition `$exists(message.body.username)` on the parsed content.

The following JSONata functions are supported:

- Aggregation functions:
  - `$average(array)`
  - `$max(array)`
  - `$min(array)`
  - `$sum(array)`
- Array functions:
  - `$append(array1, array2)`
  - `$count(array)`
  - `$reverse(array)`
  - `$sort(array [, function])`
  - `$zip(array1, ...)`
- Boolean functions:
  - `$boolean(arg)`
  - `$exists(arg)`
  - `$not(arg)`
- Numeric functions:
  - `$abs(number)`
  - `$ceil(number)`
  - `$floor(number)`
  - `$formatBase(number [, radix])`
  - `$number(arg)`
  - `$power(base, exponent)`
  - `$round(number [, precision])`
  - `$sqrt(number)`
- Object functions:
  - `$keys(object)`
  - `$lookup(object, key)`
  - `$merge(array<object>)`
  - `$spread(object)`
  - `$type(value)`  
In addition to the standard values returned by `$type()`, the API gateway implementation of `$type()` can return the following values: `binary`, `empty`, `graphql`, `stream`, or `xml`.
- String functions:
  - `$contains(str, pattern)`
  - `$join(array[, separator])`
  - `$length(str)`
  - `$lowercase(str)`
  - `$match(str, pattern [, limit])`
  - `$pad(str, width [, char])`
  - `$replace(str, pattern, replacement [, limit])`
  - `$split(str, separator [, limit])`
  - `$string(arg)`
  - `$substring(str, start[, length])`
  - `$substringAfter(str, chars)`
  - `$substringBefore(str, chars)`
  - `$trim(str)`
  - `$uppercase(str)`
- You can use the following functional extensions to standard JSONata notation. Each extension corresponds to a part of the API context.

Table 1. Functional extensions to JSONata

Extension	Variable	Description
<code>\$header(name)</code>	<code>message.headers.name</code>	Message header
<code>\$httpVerb()</code>	<code>request.verb</code>	HTTP method of the request
<code>\$operationID()</code>	<code>api.operation.id</code>	ID of the operation

Extension	Variable	Description
<code>\$operationPath()</code>	<code>api.operation.path</code>	Path of the operation
<code>\$queryParameter('name')</code>	<ul style="list-style-type: none"> <li><code>request.parameters.name.locations</code> The supported keyword is <code>query</code>.</li> <li><code>request.parameters.name.values</code></li> </ul>	Searches for the index of <code>query</code> in <code>request.parameters.name.locations</code> and returns <code>request.parameters.name.values[index]</code> , where <code>[index]</code> is the value for <code>query</code> in locations. Parameter values are not URL decoded.
<code>\$statusCode()</code>	<code>message.status.code</code>	Status code
<code>\$storageType([arg])</code>	<code>variable.body</code> You can specify any variable in the API context. When no variable is specified, the default variable <code>message.body</code> is used.	Storage type of the message. The supported values are <code>binary</code> , <code>empty</code> , <code>graphql</code> , <code>json</code> , <code>stream</code> , or <code>xml</code> .
<code>\$urlParameter('name')</code>	<ul style="list-style-type: none"> <li><code>request.parameters.name.locations</code> The supported keywords are <code>path</code> and <code>query</code></li> <li><code>request.parameters.name.values</code></li> </ul>	Searches for the index of <code>path</code> and <code>query</code> in <code>request.parameters.name.locations</code> and returns a single array that contains both <code>path</code> and <code>query</code> values from <code>request.parameters.name.values</code> . When the URL contains both path and query parameter values, the array includes the path values first followed by the query values. The values of each parameter type are added in the order that they are received. Parameter values are URL decoded. For example, the following URL contains both path and query parameter values. <code>http://example.com/petstore/cats/adopt?breed=Sphynx&amp;breed=Siamese</code> The <code>\$urlParameter('breed')</code> URL returns the following array of values. <code>[cats, adopt, Sphynx, Siamese]</code> In this example, the URL includes an API path that is configured as <code>/petstore/{breed}/{breed}</code> , where <code>breed</code> is configured to be a path parameter of the API path. As a result, <code>cats</code> and <code>adopt</code> are included in the output.
<code>\$xpath(path, xpathExpression)</code>	You can specify any writable variable in the API context. The <code>xpathExpression</code> must be a literal string.	Allows use of XPath expressions. The following example specifies all price elements in the source. <code>\$xpath(\$, '//price')</code>

Table 2. Functional extensions to JSONata for GraphQL

Extension	Variable	Description
<code>\$graphqlActiveOperation([graphql_message])</code>	<code>message.body</code>	Gets the active operation found in the specified GraphQL message. The <code>operationName</code> must be the same as the name of the active operation.
<code>\$graphqlAlias(graphql_field_node)</code>	<code>message.body</code>	Gets the alias of a GraphQL field node.
<code>\$graphqlFragments([graphql_message])</code>	<code>message.body</code>	Gets the fragments found in the specified GraphQL message.
<code>\$graphqlName([graphql_node])</code>	<code>message.body</code>	Gets the node name. For operations, the node name is the <code>operationName</code> . For fields, fragment definitions, arguments, and other elements, the node name is the name of the element. By default, the <code>operationName</code> of <code>message.body</code> is retrieved.
<code>\$graphqlOperations([graphql_message])</code>	<code>message.body</code>	Gets the operations found in the specified GraphQL message.
<code>\$graphqlType([graphql_node])</code>	<code>message.body</code>	The operation type of the active operation is retrieved for Query, Mutation, and Subscription query types.

- You can use the `&` (concatenation) navigation operator.

You can use the following JSONata numeric operators:

- `+` (addition)
- `-` (subtraction)
- `*` (multiplication)
- `/` (division)
- `%` (modulo)

You can use the following JSONata comparison operators for number values or strings:

- `=`
- `!=`
- `<`
- `>`
- `<=`
- `>=`

You can also use the following operators and expressions.

- Parentheses to convert a sequence into an array, specify operator precedence, or compute complex expressions on a context value.
- Array ranges and predicate expressions.
- Single asterisk (`*`) and double asterisk (`**`) wildcard characters.

The following elements of a GraphQL query can be exposed in JSONata notation using the syntax shown.

- `query`  
The entire GraphQL query including operations and fragments.

- **operationName**  
For an anonymous operation, **operationName** can be empty.
- **~fragmentSpreadName**
- **on~typeCondition**
- **~~fragmentDefinitionName**

You can also use the following functional extensions to standard JSONata notation. Each extension corresponds to a part of the API context.

Extension	Variable	Description
<code>\$header (name)</code>	<code>message.headers.name</code>	Message header
<code>\$httpVerb ()</code>	<code>request.verb</code>	HTTP method of the request
<code>\$operationID ()</code>	<code>api.operation.id</code>	ID of the operation
<code>\$operationPath ()</code>	<code>api.operation.path</code>	Path of the operation
<code>\$queryParameter ('name')</code>	<ul style="list-style-type: none"> <li>• <code>request.parameters.name.locations</code></li> <li>• <code>request.parameters.name.values</code></li> </ul>	Returns the value of a request query parameter given the parameter name. The function searches the <code>request.parameters.name.locations</code> array for the index position of the value <code>query</code> , and returns <code>request.parameters.name.values[index]</code> , where <code>index</code> is the index position. Parameter values are not URL decoded.
<code>\$statusCode ()</code>	<code>message.status.code</code>	Status code
<code>\$storageType ([arg])</code>	<p><b>variable.body</b> You can specify any variable in the API context. When no variable is specified, the default variable <code>message.body</code> is used.</p>	Storage type of the message. The supported values are: <ul style="list-style-type: none"> <li>• <code>binary</code></li> <li>• <code>json</code></li> <li>• <code>stream</code></li> <li>• <code>xml</code></li> </ul>
<code>\$urlParameter ('name')</code>	<ul style="list-style-type: none"> <li>• <code>request.parameters.name.locations</code></li> <li>• <code>request.parameters.name.values</code></li> </ul>	<p>Given a request parameter name, returns the parameter values for all occurrences of that parameter as a path parameter or query parameter. The function searches the <code>request.parameters.name.locations</code> array for the index positions of the values <code>path</code> and <code>query</code>, and returns, in a single array, the <code>request.parameters.name.values[index]</code> values for all identified index positions. When the URL contains both path and query parameter values, the array includes the path values first followed by the query values. The values of each parameter type are added in the order that they are received. Parameter values are URL decoded.</p> <p>For example, the following URL contains both path and query parameter values: <code>http://example.com/petstore/cats/adopt?breed=Sphynx&amp;breed=Siamese</code></p> <p>The function call <code>\$urlParameter ('breed')</code> returns the following array of values: <code>[cats, adopt, Sphynx, Siamese]</code></p> <p>In this example, the URL includes an API path that is configured as <code>/petstore/{breed}/{breed}</code>, where <code>breed</code> is configured to be a path parameter of the API path. As a result, <code>cats</code> and <code>adopt</code> are included in the output.</p>
<code>\$xpath (path, xpathExpression)</code>	You can specify any writable variable in the API context. The <code>xpathExpression</code> must be a literal string.	Allows use of XPath expressions

## Simple condition statements

The following examples show conditions that use a single function.

This example uses the `$httpVerb ()` extension to specify the HTTP method of the request.

```
$httpVerb ()="GET"
```

This example uses the `$operationPath ()` extension to specify the path of the operation.

```
$operationPath ()="/base/path-2"
```

This example uses the `$operationID ()` extension to specify the operation ID.

```
$operationID ()="test-gatewayscript-GET"
```

This example uses the `$statusCode ()` extension to specify the message status code.

```
$statusCode ()=200
```

This example uses the `$header (name)` extension to specify the content type of the message header.

```
$header ("Content-Type")="application/json"
```

## Combining conditions with logical operators

You can use `and` and `or` operators to combine multiple functions in a single condition.

This example specifies an HTTP GET request and an API operation path equal to `test-gatewayscript-GET`.

```
$httpVerb()="GET" and $operationPath()="test-gatewayscript-GET"
```

This example specifies either a **POST** or **PUT** request.

```
$httpVerb()="POST" or $httpVerb()="PUT"
```

This example specifies an API operation ID equal to **test-gatewayscript\_POST** and a message status code equal to **200**, or an API operation ID equal to **test-gatewayscript-GET** and a message status code equal to **500**.

```
($operationID()="test-gatewayscript-POST" and $statusCode()=200) or ($operationID()="test-gatewayscript-GET" and $statusCode()=500)
```

This example specifies an API operation ID equal to **test-gatewayscript-POST** and a message status code equal to **200** with an API operation path equal to **/base/path-2**.

```
($operationID()="test-gatewayscript-POST" and $statusCode()=200) and $operationPath()="/base/path-2"
```

This example specifies a message header content type equal to **text/plain** and a message header length equal to **300**, or a message status code equal to **200**.

```
($header("Content-Type")="text/plain" and $header("Content-Length")=300) or $statusCode()= 200
```

## Example switch policy

```
- switch:
 version: 2.0.0
 title: switch
 case:
 - condition: ($statusCode() = 200)
 execute:
 - invoke:
 title: invoke
 timeout: 60
 verb: GET
 cache-response: protocol
 cache-ttl: 900
 target-url: 'https://example.com/1'
 - condition: ($statusCode() != 200 and ($httpVerb() = 'GET' or $httpVerb() = 'PUT'))
 execute:
 - invoke:
 title: invoke
 timeout: 60
 verb: GET
 cache-response: protocol
 cache-ttl: 900
 target-url: 'https://example.com/2'
 - otherwise:
 - set-variable:
 title: set-variable
 actions:
 - set: message.body
 value: Default result
 description: 'Set the default result for the otherwise case'
```

DataPower Gateway (v5 compatible) only

## Writing switch condition scripts - DataPower Gateway (v5 compatible)

You write condition scripts for the **switch** policy in the DataPower® Gateway (v5 compatible) by using GatewayScript.

To reference variables in your **switch** conditions, use the form `apim.getvariable('context.location.variable')`, where *context* is the context that you want to reference, *location* is the location of the variable within that context, and *variable* is the name of the variable.

to define a **switch** condition based on the operation called, use one of the following forms:

```
- condition: "((request.verb=='GET') &&(api.operation.path=='/path-1'))"
```

```
- condition: "((api.operation.id=='Operation_ID'))"
```

where the context variables `request.verb` and `api.operation.path` retrieve the HTTP verb and the path segment of the API and operation that you want your case to apply to, while `api.operation.id` retrieves the operation ID of your operation, if one has been specified.

For information about the context variables you can use and how to reference them in your script, see [API Connect context variables](#).

## Example switch policy

```
- switch:
 version: 1.0.0
 title: switch
 case:
 - condition: message.status.code===200
 execute:
 - invoke:
 title: invoke
 timeout: 60
 verb: GET
 cache-response: protocol
 cache-ttl: 900
 target-url: 'https://example.com/1'
```

```

- condition: ((request.verb==='GET') &&(api.operation.path===' /path-2')) || ((request.verb==='GET') &&
(api.operation.path===' /path-1'))
 execute:
 - invoke:
 title: invoke
 timeout: 60
 verb: GET
 cache-response: protocol
 cache-ttl: 900
 target-url: 'https://example.com/2'
 - otherwise:
 - set-variable:
 title: set-variable
 actions:
 - set: message.body
 value: Default result
 description: Set the default result for the otherwise case

```

## throw

Use the throw construct to throw an error when it is reached during an assembly flow, usually as a result of a condition being reached.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.3.0 or later)

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [throw](#).

## About

The throw policy has the following format:

```



- throw:
 title: title
 name: 'error_name'
 error-status-code: http_status_code_for_error
 error-status-reason: http_reason_phrase_for_error
 message: error_message

```

When the throw policy is encountered, the specified error and error message is returned. If a catch has been configured that the error produced by the throw policy fulfills, the catch will be triggered.

## Properties

Table 2. throw policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
name	Yes	The name of the error to be thrown.	string
 error-status-code (policy version 2.1.0 and later)	No	Specify the HTTP status code for the error. You can use the <code>\$(variable)</code> format to reference the <code>message.status.code</code> API context variable.	integer
 error-status-reason (policy version 2.1.0 and later)	No	Specify the HTTP reason phrase for the error. You can use the <code>\$(variable)</code> format to reference the <code>message.status.reason</code> API context variable.	string
message	No	The message to accompany the error.	string

## Example

```

- throw:
 version: 1.0.0
 title: throw
 name: '404'
 message: Not found

```

## User-defined policies

You can apply your own user-defined policies to your APIs.

The policy configuration that you add to the OpenAPI definition file must conform to the schema that you defined in the YAML file that describes the policy, and will have the following general format:

```

policy_name:
 property1: value1
 property2: value2
 .
 .
 .

```

For details on user-defined policies, and how to describe and implement them, see [Authoring policies](#).

## user-security

Use the user-security policy to extract a user's credentials, authenticate those credentials, and obtain authorization from the user.

### Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [User Security](#).

### About

The user-security policy has the following format:

```

- user-security:
 version: version
 title: title
 description: description
 factor-id: factor ID
 extract-identity-method: method_used_to_extract_credentials
 .
 .
 .
 properties_specific_to_the_specified_identity_extraction_method
 .
 .
 .
 user-auth-method: authentication_method
 .
 .
 .
 properties_specific_to_the_specified_authentication_method
 .
 .
 .
 user-az-method: authorization_method
 .
 .
 .
 properties_specific_to_the_specified_authorization_method
 .
 .
 .

```

### Properties

Table 2. user-security policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	The title of the policy.	string
description	No	A description of the policy.	string
factor-id	No	The identity that identifies the results of factor-authentication in the API context.	string

Property	Required	Description	Data type
extract-identity-method	Yes	<p>Select the method that is used to extract the user credentials. The following options are available:</p> <p><b>basic</b> Use basic authentication; no additional configuration is required.</p> <p><b>context-var</b> The credentials are provided by API Connect context variables; specify the following properties:</p> <ul style="list-style-type: none"> <li>• user-context-var: the context variable that is used to obtain the user name.</li> <li>• pass-context-var: the context variable that is used to obtain the password.</li> </ul> <p><b>html-form</b> Use forms based identity-extraction. You can use the default form or a custom form. To use the default form, specify <b>ei-default-form: true</b>. To use a custom form, specify <b>ei-default-form: false</b> and supply the following properties:</p> <ul style="list-style-type: none"> <li>• ei-custom-form: the location of the form.</li> <li>• ei-custom-form-tls-client-profile: the TLS client profile that is used to secure the connection to the remote server.</li> </ul> <p>Use the ei-form-time-limit property to specify the time allowed to submit the form.</p> <p><b>redirect</b> Use a redirect for identity-extraction; specify the following properties:</p> <ul style="list-style-type: none"> <li>• redirect-url: the URL fragment to which to redirect the request to obtain user credentials.</li> <li>• redirect-time-limit: the time allowed for the transaction to complete.</li> </ul> <p><b>disabled</b> Identity-extraction is disabled; this aspect of processing is skipped.</p> <p>Specify <b>ei-stop-on-error: true</b> to halt assembly processing in the event of identity-extraction failure. This field is required if identity extraction is not disabled.</p>	string
user-auth-method	Yes	<p>Select the authentication method. The following options are available:</p> <p><b>auth-url</b> The credentials are authenticated by an external endpoint; specify the following properties:</p> <ul style="list-style-type: none"> <li>• auth-url: the URL of the authentication endpoint.</li> <li>• auth-tls-client-profile: the TLS client profile that is used to secure the connection to the authentication endpoint.</li> <li>• auth-response-headers-pattern: the pattern that is used to select which response headers to add to the API context.</li> <li>• auth-response-header-credential: the response header that contains the authenticated user credentials.</li> </ul> <p><b>ldap</b> The credentials are authenticated by an LDAP user registry; use the ldap-registry property to specify the required registry.</p> <p><b>disabled</b> Authentication is disabled; this aspect of processing is skipped.</p> <p>Specify <b>au-stop-on-error: true</b> to halt assembly processing in the event of user authentication failure. This field is required if authentication is not disabled.</p>	string
user-az-method	Yes	<p>Select the authorization method. The following options are available:</p> <p><b>authenticated</b> Implicitly accept any previously authenticated users; no additional configuration is required.</p> <p><b>html-form</b> The user provides authorization through an HTML form. You can use the default form or a custom form. To use the default form, specify <b>az-default-form: true</b>. To use a custom form, specify <b>az-default-form: false</b>.</p> <p>Supply the following properties:</p> <ul style="list-style-type: none"> <li>• az-table-dynamic-entries: the name of a context variable that specifies the scopes that are to be added automatically to the authorization consent form.</li> <li>• az-form-time-limit: the time allowed to submit the form.</li> </ul> <p>For a custom form, supply the following properties:</p> <ul style="list-style-type: none"> <li>• az-custom-form: the location of the form.</li> <li>• az-custom-form-tls-client-profile: the TLS client profile that is used to secure the connection to the remote server.</li> </ul> <p><b>disabled</b> Authorization is disabled; this aspect of processing is skipped.</p> <p>Specify <b>az-stop-on-error: true</b> to halt assembly processing in the event of user authorization failure. This field is required if authorization is not disabled.</p>	string

## Example

```
basic authentication with LDAP registry
```

```
- user-security:
 version: 2.0.0
 title: user-security
 factor-id: default
 extract-identity-method: basic
 ei-stop-on-error: true
 user-auth-method: ldap
 ldap-registry: corporate-ldap
 au-stop-on-error: true
 user-az-method: authenticated
 az-stop-on-error: true
```



## validate - DataPower API Gateway

Use the validate policy to validate the payload in an assembly flow against a schema.

## Gateway support

Note: This page describes the validate policy implementation in the DataPower® API Gateway. If you are using the DataPower Gateway (v5 compatible), see [validate - DataPower Gateway \(v5 compatible\)](#).

For information on the different types of gateway, see [API Connect gateway types](#).

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower API Gateway	2.0.0
	2.1.0 (DataPower API Gateway Version 10.0.1.0 or later)
	2.2.0 (DataPower API Gateway Version 10.0.2.0 or later)
	2.3.0 (DataPower API Gateway Version 10.0.2.0 or later)
	2.4.0 (DataPower API Gateway Version 10.0.4.0 or later)
	2.5.0 (DataPower API Gateway Version 10.5.0.0 or later)
	2.6.0 (DataPower API Gateway Version 10.5.0.2 or later)
	2.7.0 (DataPower API Gateway Version 10.5.0.3 or later)

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Validate - DataPower API Gateway](#).

## About

The validate policy has the following format:

```
- validate:
 version: version
 title: title
 description: description
 validate-against: validation_mechanism
 .
 .
 .
 properties_specific_to_the_specified_validation_mechanism
 .
 .
```

Apply this policy by adding an assembly extension with an execute field to your OpenAPI definition file.

Note: With the DataPower API Gateway, the input to the validate policy must be parsed data. One way to produce parsed data is to use a [parse](#) policy before a validate policy in your assembly flow, which provides explicit control of the parse action.

## Properties

The following table describes the policy properties:

Table 2. validate policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string



Property	Required	Description	Data type
input	No	Identifies a variable in the API context. The content of the <b>body</b> field of the variable, which is represented by <b>variable_name.body</b> , is the input data to validate. By default, the variable name is <b>message</b> .	string
output	No	Specifies the name of a variable in the API context. <ul style="list-style-type: none"> <li>If the validation passes, the body field of the output variable, which is represented by <b>variable_name.body</b>, stores the output of the assembly validate action.</li> <li>If the schema to validate is a JSON schema, the validation also adds any default values that are missing from the payload.</li> <li>If the validation fails, no output is stored.</li> <li>If an output variable is not specified, the results of the assembly validate action are not saved.</li> </ul>	string
validate-against	Yes	Specifies the schema to be used for validating the payload. Valid values: <ul style="list-style-type: none"> <li><b>definition</b>: A previously defined schema will be used to validate the payload that is returned from other invoke actions or tasks in the assembly flow. In addition, supply a <b>definition</b> property to specify the required schema.</li> <li><b>url</b>: the schema is identified by a URL location. In addition, supply the following properties: <ul style="list-style-type: none"> <li><b>json-schema</b> field: the URL of the JSON schema to be used for validating a JSON payload.</li> <li><b>xml-validation-mode</b>, specify one of the following values to define how an XML payload is validated: <ul style="list-style-type: none"> <li><b>xsd</b>: use an XML schema; in addition, supply an <b>xml-schema</b> property that specifies the URL of the XML schema.</li> <li><b>wSDL</b>: use a WSDL schema; in addition, supply an <b>xml-schema</b> property that specifies the URL of the WSDL schema to be used for validating a SOAP payload..</li> <li><b>soap-body</b>: validate the body of a SOAP message against the XML schema only.</li> </ul> </li> </ul> </li> </ul> <p>Note: The following limitations apply to schemas used for JSON validation, which include JSON schemas and OpenAPI documents that are used as schemas for validation. Exceeding these limits can impact compilation performance and is not supported.</p> <ul style="list-style-type: none"> <li>Maximum of 6,500 lines. Each key and each item in an array count as a line.</li> <li>Maximum recursion depth of 250.</li> <li>Maximum of 3,000 items in <b>enum</b> lists.</li> </ul> <ul style="list-style-type: none"> <li><b>wSDL</b>: (use with a SOAP service based API only) use the XML schema in the WSDL file associated with the API operation or the API path.</li> <li><b>body-param</b>: validate the request input against the schema definition that is specified in the <b>schema</b> property for the request parameter for this operation.</li> <li><b>response-param</b>: validate the response to be returned to the client application, against the schema definition that is specified in the <b>schema</b> property for the response parameter for this operation.</li> <li><b>graphql</b>: (use with a GraphQL proxy API only) validate the payload against the GraphQL schema that has been imported into the GraphQL proxy API. In addition, either the GraphQL query or response, depending on the input, is analyzed against the GraphQL schema to calculate the cost, and the result is placed in the API context.</li> </ul> <p>For more information on GraphQL proxy APIs, see <a href="#">Creating a GraphQL proxy API</a> and <a href="#">GraphQL context variables</a>.</p>	string
xslt-version	No	The XSLT processor version. The default value is XSLT10.	string
strict	No	Whether to enable strict XSLT error checking. Non-strict operations attempt to recover from certain errors, such as use of undeclared variables, calling undeclared templates, and so forth. By default, strict XSLT error checking is enabled.	boolean
profile	No	Whether to enable stylesheet profiling. This option should not be used in production environments. By default, stylesheet profiling is disabled.	boolean
debug	No	Whether to run the stylesheet, XQuery script, and JSONiq script in debug mode. When a stylesheet, XQuery script, or JSONiq script is run in debug mode, it generates a custom web page instead of displaying its normal output. The web page details exactly what occurred during execution, including the values of variables and where particular pieces of the output came from. This option should not be used in production environments. By default, debug mode is disabled.	boolean
stream	No	Whether the stylesheet must be run in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, an error is generated and the input is not processed. By default, streaming mode is disabled.	boolean
try-stream	No	Whether to attempt to run the stylesheet in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, a warning is generated during compilation and the stylesheet is read in the entire input as normal at execution time. By default, attempting to run the stylesheet in streaming mode is disabled.	boolean
minimum-escaping	No	Whether to escape output produced from the stylesheet during processing. Minimal escaping is particularly useful when handling non-English character sets. By default, minimum escaping is disabled.	boolean
stack-size	No	The maximum number of bytes that the stack is allowed to use while executing a stylesheet or other compiled content. This setting is used to block infinite recursion. The minimum value is 10 kilobytes, or 10,240 bytes. The maximum value is 100 megabytes, or 104,857,600 bytes. The default value is 1 megabyte, or 1,048,576 bytes.	integer

Property	Required	Description	Data type
ws-i-validation	No	The validation behavior to apply to WSDL files that are checked for conformance to section 5 of WS-I Basic Profile (version 1.0, April 2004). The default setting is Warn.  Ignore Disables conformance checking. Warn Logs warnings for violations. Fail Forces conformance. Fails if the file contains any violation.	string
wsdl-validate-body	No	The validation behavior for the <b>soap:Body</b> . The default setting is Strict.  Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition. Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match. Skip Disables validation of this part of the message.	string
wsdl-validate-headers	No	The validation behavior for the <b>soap:Header</b> . The default setting is Lax.  Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition. Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match. Skip Disables validation of this part of the message.	string
wsdl-validate-faults	No	Specifies the validation behavior for the fault detail. The default setting is Strict.  Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition. Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match. Skip Disables validation of this part of the message.	string
wsdl-wrapped-faults	No	Whether to require compatibility with RPC-style wrappers. By default, RPC-style wrappers are not required.	boolean
allow-soap-enc-array	No	Whether to allow the schema to accept most uses of elements with <b>xsi:type='SOAP-ENC:Array'</b> consistent with SOAP 1.1 Section 5, even when these attributes violate the XML Schema specification. Normally, the <b>xsi:type</b> attribute must name a type equal to or derived from the actual type of the element. For schemas compiled with this option, <b>xsi:type</b> is accepted specifically for the SOAP 1.1 Encoding 'Array' complex type if the element type is derived from <b>SOAP-ENC:Array</b> . The opposite is the normal allowable case. By default, elements with <b>xsi:type='SOAP-ENC:Array'</b> are not accepted.	boolean
validate-soap-enc-array	No	Whether to perform extra schema validation following the encoding rules in SOAP 1.1 Section 5. When enabled, members of SOAP arrays are validated, attributes such as @id and @href are allowed even if they are not allowed by the schema, and @href values are checked to ensure that they have a corresponding @id element. By default, the extra validation is not performed.	boolean
wildcards-ignore-xsi-type	No	Whether <b>xs:any</b> elements in the schema validate only child elements by name. The XML Schema specification requires that, if a wildcard matches an element but that element does not have an element declaration, the element is instead validated according to an <b>xsi:type</b> attribute on it. This option ignores those <b>xsi:type</b> attributes. It should be used for cases such as SOAP envelope validation where a further validation step will validate the contents matching the wildcard, possibly using the SOAP 1.1 encoding rules. By default, <b>xsi:type</b> attributes are not ignored.	boolean
wsdl-strict-soap-version	No	Whether to strictly follow the SOAP binding in the WSDL. When enabled, only messages bound to SOAP 1.2 appear in SOAP 1.2 envelopes and only messages bound to SOAP 1.1 appear in SOAP 1.1 envelopes. By default, strict SOAP binding is disabled.	boolean
xacml-debug	No	Whether to compile XACML policies with debug information. Note that the XACML debugging messages are also controlled by the log event in the XACML category. Use the debug log level to view the full XACML debugging messages. By default, XACML policies are not compiled with debug information.	boolean

Property	Required	Description	Data type
allow-xop-include	No	<p>Specifies whether the schema or WSDL document accepts messages where base64-encoded binary content was optimized according to the MTOM/XOP specifications. XOP binary-optimization replaces base64-encoded binary data with an <b>xop:Include</b> reference element that references the unencoded binary data located in an attachment. By default, MTOM/XOP optimized messages are disabled.</p> <ul style="list-style-type: none"> <li>When disabled, such optimized messages are rejected by validation of the optimized form. Rejection occurs because the schema specifies a simple type that accepts base64-encoded data, such as <b>xs:base64Binary</b> or <b>xs:string</b>, but the message contains an <b>xop:Include</b> element instead.</li> <li>When enabled, an <b>xop:Include</b> element can optionally appear in place of content for any XML Schema simple type that validates base64-encoded binary data. The <b>xop:Include</b> element itself will be validated according to the built-in schema in <b>store:///schemas/xop.xsd</b>.</li> </ul>	boolean

You can also apply a validate policy by using the API Designer assembly editor to add a built-in policy to the API. For more information, see [Validate - DataPower API Gateway](#) in the built-in policies section.

## Example 1

```
- validate:
 version: 2.0.0
 title: 'validate, response parameter schema'
 validate-against: response-param
```

## Example 2

```
- validate:
 version: 2.0.0
 title: 'validate, predefined schema'
 validate-against: definition
 definition: '#/definitions/RouteOutput'
```

## Example 3

```
- validate:
 version: 2.0.0
 title: 'validate, JSON and XML schema URLs'
 validate-against: url
 json-schema: 'https://my.json.schema'
 xml-validation-mode: xsd
 xml-schema: 'https://my.xml.schema'
```

DataPower Gateway (v5 compatible)

## validate - DataPower Gateway (v5 compatible)

Use the validate policy to validate the payload in an assembly flow against a schema.

## Gateway support

Note: This page describes the validate policy implementation in the DataPower® Gateway (v5 compatible). If you are using the DataPower API Gateway, see [validate - DataPower API Gateway](#).

For information on the different types of gateway, see [API Connect gateway types](#).

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower Gateway (v5 compatible)	1.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Validate - DataPower Gateway \(v5 compatible\)](#).

Restriction:

- The schema that represents the XML can reference only one XML namespace.
- The schema cannot reference polymorphic XML elements.
- The validation works on the **message.body** variable and not any other output/context variable. If the invoke policy contains a configured response object variable, then **message.body** is not set, and validate is not able to act.
- If you use the **multipleOf** keyword in a schema definition for the API then, due to rounding behavior, the specified value must satisfy the following conditions, otherwise the validation fails when the API is called:
  - The value must not be less than 0.0000009999999999999999848869.
  - If the value is greater than 1, the amount before the decimal point must not be greater than 999999999999999934463.

## About

The validate policy has the following format:

```
- validate:
 version: version
 title: title
 description: description
 definition: swagger_schema_definition_to_be_used
```

Apply this policy by adding an assembly extension with an execute field to your OpenAPI definition file.

## Properties

The following table describes the policy properties:

Table 2. validate policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
definition	Yes	The schema to be used to validate the payload. Valid values: <ul style="list-style-type: none"> <li><b>request</b>: Select this value to validate the request input against the schema definition that is specified in the Type field for the request parameter for this operation. For information about how to create a request parameter, see <a href="#">Configuring an operation</a>.</li> <li><b>response</b>: Select this value to validate the response to be returned to the client application, against the schema definition that is specified in the Schema field for the response parameter for this operation. For information about how to create a response parameter, see <a href="#">Configuring an operation</a>.</li> <li>The name of a schema definition, in the following format:               <pre>#/definitions/schema_name</pre>               The schema must be defined in the <b>definitions</b>: section of your OpenAPI file.             </li> </ul>	string

You can also apply an validate policy by using the API Designer assembly editor to add a built-in policy to the API. For more information, see [Validate - DataPower Gateway \(v5 compatible\)](#), in the built-in policies section.

## Example 1

```
validate:
 version: 1.0.0
 title: validate the response
 definition: #/definitions/RouteOutput
```

## validate-username token

Use the Validate Username Token policy to validate a Web Services Security (WS-Security) UsernameToken in a SOAP payload before allowing access to the protected resource.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0 1.1.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [Validate Username Token](#).

## About

The validate-username token policy has the following format:

```
- validate-username token:
 version: version
 title: title
 description: description
 auth-type: Authentication URL or LDAP Registry (policy version 1.0.0 only)
 auth-url: authentication_url_to_use (policy version 1.0.0 only)
 tls-profile: tls_profile_to_use (policy version 1.0.0 only)
 ldap-registry: name_of_the_ldap_user_registry (policy version 1.0.0 only)
 registry: name_of_the_ldap_or_authurl_user_registry (policy version 1.1.0 and later)
 ldap-search-attribute: name_of_the_ldap_user_password_attribute
```

## Properties

The following table describes the policy properties:

Table 2. Validate Username Token policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	Yes	The title of the policy.	string
description	No	A description of the policy.	string
auth-type (policy version 1.0.0 only)	Yes	The authentication type to use to validate the UsernameToken. Valid values: <ul style="list-style-type: none"> <li><b>Authentication URL</b>: Specify this value to validate the user credentials against an authentication URL.</li> <li><b>LDAP registry</b>: Specify this value to validate the user credentials against an LDAP user registry.</li> </ul> The default value is: <b>Authentication URL</b> .	string
auth-url (policy version 1.0.0 only)	Yes	The authentication URL to use to validate the UsernameToken user credentials against. Note: This property is required only if Authentication type is set to <b>Authentication URL</b> .	string
tls-profile (policy version 1.0.0 only)	No	The TLS profile to use for the secure transmission of data to the authentication URL. Note: This property is available only if Authentication type is set to <b>Authentication URL</b> .	string
ldap-registry (policy version 1.0.0 only)	Yes	The name of the LDAP user registry to validate the UsernameToken user credentials against. Note: This property is required only if Authentication type is set to <b>LDAP registry</b> .	string
registry (policy version 1.1.0 and later)	Yes	The name of the LDAP or Authentication URL registry to use to validate the UsernameToken.	string
ldap-search-attribute	Yes	The name of the LDAP user password attribute. Note: This property is required only for an LDAP user registry.	string

## Examples

The following example shows an LDAP user registry authentication:

```
- validate-username-token:
 version: 1.0.0
 title: "validate-username-token"
 auth-type: "LDAP Registry"
 ldap-registry: "wstest"
 ldap-search-attribute: "userPassword"
```

The following example shows an Authentication URL definition:

```
- validate-username-token:
 version: 1.0.0
 title: "validate-username-token"
 auth-type: "Authentication URL"
 auth-url: "https://www.google.com"
 tls-profile: "default-ssl-profile"
```

For more information about how to use a validate-username-token security policy, see [Validate Username Token](#) in the built-in policies section.

## websocket-upgrade

Use the websocket-upgrade policy to process API requests and responses through a WebSocket connection.

## Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [WebSocket Upgrade](#).

The websocket-upgrade policy has the following format:

```
- websocket-upgrade:
 version: version
 title: title
 description: description
 target-url: URL_of_target_API
 tls-profile: TLS_profile_to_be_used
 timeout: timeout_value_in_seconds
 follow-redirects: redirect_behavior_on_301_error
 username: username_if_authentication_required
 password: password_if_authentication_required
 inject-proxy-headers: are_proxy_headers_sent_to_target_url
 decode-request-params: are_request_parameters_decoded
 encode-plus-char: are_plus_characters_encoded
 header-control:
 .
 .
 headers_to_copy_to_target_url
 .
```

```

parameter-control:
.
.
.
parameters_to_copy_to_target_url
.
.
.

```

## Properties

Table 2. websocket-upgrade policy properties

Property	Required	Description	Data type
version	Yes	The policy version number.	string
title	No	The title of the policy.	string
description	No	A description of the policy.	string
target-url	Yes	Specify the URL to be invoked.	string
tls-profile	No	Specifies a TLS profile to use for the secure transmission of data.	string
timeout	No	The time to wait before a reply back from the endpoint (in seconds). The default value is <b>60</b> .	integer
follow-redirects	No	Specifies the behavior if the back-end server returns the HTTP status code <b>301 Moved Permanently</b> . If this property is set to <b>true</b> , the <b>invoke</b> policy follows the URL redirection by making a further call to the URL specified in the <b>Location</b> header in the response. If this property is set to <b>false</b> , the <b>invoke</b> saves the <b>301</b> status code and the API call is considered to be complete. Note: The <b>follow-redirect</b> property is supported only by the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), the <b>invoke</b> always follows the URL redirection; the <b>proxy</b> policy (not supported by the DataPower API Gateway) saves the <b>301</b> status code and completes the API call without following the URL redirection.	boolean
username	No	The username to use for HTTP Basic authentication.	string
password	No	The password to use for HTTP Basic authentication.	string
inject-proxy-headers	No	When set to true, the <b>invoke</b> policy injects the <b>X-Forwarded-For</b> , <b>X-Forwarded-To</b> , <b>X-Forwarded-Host</b> , and <b>X-Forwarded-Proto</b> headers to the request that is sent to the <b>target-url</b> . The default value is <b>false</b> .	boolean
decode-request-params	No	When set to true, any request parameters that are referenced by a variable definition on the <b>target-url</b> of the <b>invoke</b> policy are URL-decoded. The default value is <b>false</b> .	boolean
encode-plus-char	No	When set to true, all "+" characters in the query parameter values of the <b>target-url</b> are encoded to %2F. The default value is <b>false</b> .	boolean
<b>header-control:</b> <b>type</b> <b>values</b>	No	Specifies the headers in <b>message.headers</b> that you want to copy to the target URL. If the <b>type</b> property is set to <b>blocklist</b> , the <b>values</b> property lists the headers that you don't want to be copied. If the <b>values</b> property is empty then all headers are copied.  If the <b>type</b> property is set to <b>allowlist</b> , the <b>values</b> property lists the headers that you want to be copied.  The items that are listed in the <b>values</b> property are in regular expression format. The values are not case-sensitive. The default value of the <b>header-control</b> property is  <b>header-control:</b> <b>type: blocklist</b> <b>values: []</b>  See <a href="#">header-control examples</a>	object
<b>parameter-control:</b> <b>type</b> <b>values</b>	No	Specifies the parameters in the incoming request that you want to be copied to the target URL. If the <b>type</b> property is set to <b>blocklist</b> , the <b>values</b> property lists the parameters that you don't want to be copied.  If the <b>type</b> property is set to <b>allowlist</b> , the <b>values</b> property lists the parameters that you want to be copied. If the <b>values</b> property is empty then no parameters are copied.  The items that are listed in the <b>values</b> property are in regular expression format. The values are not case-sensitive. The default value of the <b>parameter-control</b> property is  <b>parameter-control:</b> <b>type: allowlist</b> <b>values: []</b>  See <a href="#">parameter-control examples</a>	object
<b>request-assembly:</b> <b>execute</b> . . . <b>policy</b> <b>assembly</b> . . .	No	The request processing assembly. For details on configuring an assembly in the <b>execute</b> section, see <a href="#">execute</a> .	object

Property	Required	Description	Data type
response-assembly: execute . . . policy assembly . . .	No	The response processing assembly. For details on configuring an assembly in the <code>execute</code> section, see <a href="#">execute</a> .	object

## Example

```
- websocket-upgrade:
 version: 2.0.0
 title: websocket-upgrade
 timeout: 60
 target-url: 'https://my.websocket.upgrade.com'
 secure-gateway: false
 follow-redirects: true
 inject-proxy-headers: true
 decode-request-params: false
 encode-plus-char: true
 response-assembly:
 execute:
 - parse:
 version: 2.0.0
 title: parse
 parse-settings-reference:
 default: apic-default-parsesettings
 description: Parse the response from the backend server
```

## header-control examples

```
copy all headers to the target URL

- invoke:
 target-url: http://myhost/mypath
 header-control:
 type: blacklist
 values: []

copy all headers except X-Client-ID and Content-Type

- invoke:
 target-url: http://myhost/mypath
 header-control:
 type: allowlist
 values:
 - ^X-Client-ID$
 - ^Content-Type$

copy no headers

- invoke:
 target-url: http://myhost/mypath
 header-control:
 type: allowlist
 values: []

copy only the Content-Type header

- invoke:
 target-url: http://myhost/mypath
 header-control:
 type: allowlist
 values:
 - ^Content-Type$
```

## parameter-control examples

```
copy no request parameters to the target URL

- invoke:
 target-url: http://myhost/path?storeid=3
 parameter-control:
 type: allowlist
 values: []

append the petid parameter to the target URL
if the incoming request is http://apigw/org/sandbox/petstore/base?petid=100&display=detailed,
the target URL at runtime will be http://myhost/mypath?storeid=3&petid=100

- invoke:
 target-url: http://myhost/path?storeid=3
 parameter-control:
 type: allowlist
 values:
 - ^petid$
```

## xml-to-json

Use the xml-to-json policy to convert the context payload of your API from the extensible markup language (XML) format to JavaScript Object Notation (JSON).

### Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [XML to JSON](#).

### About

The xml-to-json policy has the following structure:

```
- xml-to-json:
 version: version
 title: Title
 description: Description
```

Note: If you are using the DataPower API Gateway, the input to the xml-to-json policy must be parsed data. One way to produce parsed data is to use a [parse](#) policy before an xml-to-json policy in your assembly flow, which provides explicit control of the parse action.

### Properties

The following table describes the policy properties:

Table 2. Policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	Yes	The title of the policy.	string
description	No	A description of the policy.	string
<span>API Gateway only</span> input	No	The input message to convert. Specify the name of a variable in the API context. <b>variableName.body</b> , the message payload, represents the JSON input to convert. The default value of the variable is <b>message</b> and <b>message.body</b> is the default input.	string
<span>API Gateway only</span> output	No	The output message to store the conversion result. Specify the name of a variable in the API context. <b>variableName.body</b> represents the result of conversion from JSON format to XML format. When the specified input message is the default message, the default output is <b>message.body</b> . Otherwise, when the input message is the variable <b>my-message-variable</b> , for example, the default output is <b>my-message-variable.body</b> . The variable cannot be any read-only in the API context.	string
<span>API Gateway only</span> conversionType	No	The conversion type that determines the target format of the output. The following options are available: <ul style="list-style-type: none"><li>badgerFish: BadgerFish convention is used to determine the target conversion format of the output.</li><li>apicv5: apicv5 convention is used to determine the target conversion format of the output.</li></ul>	string

### Example

The following is an example of a xml-to-json policy:

```
- xml-to-json:
 version: 1.0.0
 title: XML to JSON transform
 description: Transforms XML message body to JSON format
```

## xslt

Use the xslt policy to apply an XSLT transform to the payload of the API definition.

### Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.4.0 or later)

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see [XSLT](#).



## About

The xslt policy has the following structure:












```
- xslt:
 version: version
 title: Title
 description: Description
 input: Input_True_False
 source: Transform
```

Note: If you are using the DataPower API Gateway, the input to the xslt policy must be parsed data. One way to produce parsed data is to use a [parse](#) policy before an xslt policy in your assembly flow, which provides explicit control of the parse action.

## Properties

The following table describes the policy properties:

Table 2. xslt policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	Yes	The title of the policy.	string
description	No	A description of the policy.	string
input	No	Indicates whether this XSLT input document uses the context current payload, or if there is no input. The default value is <b>false</b> .	boolean
 serialize-output	No	If set to <b>true</b> , the output tree that is generated by the XSLT policy is serialized. The content of <code>message.body</code> is updated with the serialized binary data rather than the XML tree. The default value is <b>false</b> .	boolean
source	Yes	The XSLT transform source to execute.	string
 xslt-version	No	The XSLT processor version. The default value is XSLT10.	string
 strict	No	Whether to enable strict XSLT error checking. Non-strict operations attempt to recover from certain errors, such as use of undeclared variables, calling undeclared templates, and so forth. By default, strict XSLT error checking is enabled.	boolean
 profile	No	Whether to enable stylesheet profiling. This option should not be used in production environments. By default, stylesheet profiling is disabled.	boolean
 debug	No	Whether to run the stylesheet, XQuery script, and JSONiq script in debug mode. When a stylesheet, XQuery script, or JSONiq script is run in debug mode, it generates a custom web page instead of displaying its normal output. The web page details exactly what occurred during execution, including the values of variables and where particular pieces of the output came from. This option should not be used in production environments. By default, debug mode is disabled.	boolean
 stream	No	Whether the stylesheet must be run in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, an error is generated and the input is not processed. By default, streaming mode is disabled.	boolean
 try-stream	No	Whether to attempt to run the stylesheet in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, a warning is generated during compilation and the stylesheet is read in the entire input as normal at execution time. By default, attempting to run the stylesheet in streaming mode is disabled.	boolean
 minimum-escaping	No	Whether to escape output produced from the stylesheet during processing. Minimal escaping is particularly useful when handling non-English character sets. By default, minimum escaping is disabled.	boolean
 stack-size	No	The maximum number of bytes that the stack is allowed to use while executing a stylesheet or other compiled content. This setting is used to block infinite recursion. The minimum value is 10 kilobytes, or 10,240 bytes. The maximum value is 100 megabytes, or 104,857,600 bytes. The default value is 1 megabyte, or 1,048,576 bytes.	integer
 wsi-validation	No	The validation behavior to apply to WSDL files that are checked for conformance to section 5 of WS-I Basic Profile (version 1.0, April 2004). The default setting is Warn.  Ignore Disables conformance checking. Warn Logs warnings for violations. Fail Forces conformance. Fails if the file contains any violation.	string
 wsd-validate-body	No	The validation behavior for the <code>soap:Body</code> . The default setting is Strict.  Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition.  Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match.  Skip Disables validation of this part of the message.	string

Property	Required	Description	Data type
API Gateway only wsdl-validate-headers	No	The validation behavior for the <code>soap:Header</code> . The default setting is Lax.  Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition.  Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match.  Skip Disables validation of this part of the message.	string
API Gateway only wsdl-validate-faults	No	Specifies the validation behavior for the fault detail. The default setting is Strict.  Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition.  Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match.  Skip Disables validation of this part of the message.	string
API Gateway only wsdl-wrapped-faults	No	Whether to require compatibility with RPC-style wrappers. By default, RPC-style wrappers are not required.	boolean
API Gateway only allow-soap-enc-array	No	Whether to allow the schema to accept most uses of elements with <code>xsi:type='SOAP-ENC:Array'</code> consistent with SOAP 1.1 Section 5, even when these attributes violate the XML Schema specification. Normally, the <code>xsi:type</code> attribute must name a type equal to or derived from the actual type of the element. For schemas compiled with this option, <code>xsi:type</code> is accepted specifically for the SOAP 1.1 Encoding 'Array' complex type if the element type is derived from <code>SOAP-ENC:Array</code> . The opposite is the normal allowable case. By default, elements with <code>xsi:type='SOAP-ENC:Array'</code> are not accepted.	boolean
API Gateway only validate-soap-enc-array	No	Whether to perform extra schema validation following the encoding rules in SOAP 1.1 Section 5. When enabled, members of SOAP arrays are validated, attributes such as @id and @href are allowed even if they are not allowed by the schema, and @href values are checked to ensure that they have a corresponding @id element. By default, the extra validation is not performed.	boolean
API Gateway only wildcards-ignore-xsi-type	No	Whether <code>xs:any</code> elements in the schema validate only child elements by name. The XML Schema specification requires that, if a wildcard matches an element but that element does not have an element declaration, the element is instead validated according to an <code>xsi:type</code> attribute on it. This option ignores those <code>xsi:type</code> attributes. It should be used for cases such as SOAP envelope validation where a further validation step will validate the contents matching the wildcard, possibly using the SOAP 1.1 encoding rules. By default, <code>xsi:type</code> attributes are not ignored.	boolean
API Gateway only wsdl-strict-soap-version	No	Whether to strictly follow the SOAP binding in the WSDL. When enabled, only messages bound to SOAP 1.2 appear in SOAP 1.2 envelopes and only messages bound to SOAP 1.1 appear in SOAP 1.1 envelopes. By default, strict SOAP binding is disabled.	boolean
API Gateway only xacml-debug	No	Whether to compile XACML policies with debug information. Note that the XACML debugging messages are also controlled by the log event in the XACML category. Use the debug log level to view the full XACML debugging messages. By default, XACML policies are not compiled with debug information.	boolean
API Gateway only allow-xop-include	No	Specifies whether the schema or WSDL document accepts messages where base64-encoded binary content was optimized according to the MTOM/XOP specifications. XOP binary-optimization replaces base64-encoded binary data with an <code>xop:Include</code> reference element that references the unencoded binary data located in an attachment. By default, MTOM/XOP optimized messages are disabled.  <ul style="list-style-type: none"> <li>When disabled, such optimized messages are rejected by validation of the optimized form. Rejection occurs because the schema specifies a simple type that accepts base64-encoded data, such as <code>xs:base64Binary</code> or <code>xs:string</code>, but the message contains an <code>xop:Include</code> element instead.</li> <li>When enabled, an <code>xop:Include</code> element can optionally appear in place of content for any XML Schema simple type that validates base64-encoded binary data. The <code>xop:Include</code> element itself will be validated according to the built-in schema in <code>store:///schemas/xop.xsd</code>.</li> </ul>	boolean

You can also apply an xslt policy by using the API Designer assembly editor to add a built-in policy to the API. For more information, see [XSLT](#) in the built-in policies section.

For examples of the OpenAPI definitions of xslt policies, see [XSLT policy examples](#) in the built-in policies section.

For more examples of how to use XSLT to access and modify properties and context, see [Implementation code examples](#) and, if you are using the DataPower API Gateway, [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway](#).

## Related concepts

- [Variable references in API Connect](#)

## Related reference

- [XSLT](#)
- [API Connect context variables](#)

## catch

Use the `catch` extension to catch errors that occur during an API call.

The `catch` extension takes the following form:

```
catch:
 - errors:
 - Error_1
 - Error_2
 - execute:
 assembly_1
 - errors:
 - Error_3
 - execute:
 assembly_2
 - default:
 assembly_3
```

The following table shows the properties of the `catch` extension:

Table 1. The properties of the `catch` extension

Property	Required	Description	Type
<code>errors</code>	Yes	The errors for which the catch will activate. For a list of errors, see <a href="#">Error cases supported by assembly catches</a> .	Array (String)
<code>execute</code>	Yes	The section of the assembly that will execute when the catch is activated.	Array (Object)
<code>default</code>	No	The section of the assembly that will execute when an error does not trigger other catches. It behaves in the same manner and has the same structure as <code>execute</code> .	Array (Object)

The following is an example of a `catch` extension:

```
catch:
 - errors:
 - ConnectionError
 - JavaScriptError
 execute:
 - activity-log:
 title: activity-log
 content: activity
 error-content: payload
 - default:
 - activity-log:
 title: activity-log
 content: activity
 error-content: payload
```

## properties

Use the `properties` extension to define properties for referencing in an API.

The `properties` extension has the following structure:

```
properties:
 property_1:
 value: default_value_1
 description: description_1
 encoded: encoded_boolean_1
 property_2:
 value: default_value_2
 description: description_2
 encoded: encoded_boolean_2
```

The following table lists the fields found in the `properties` extension:

Table 1. The properties extension

Property	Required	Description	Data type
<code>property</code>	Yes	The name of the property. It is used when referencing the property. Note that this is the field name, not the contents of the field.	Object
<code>value</code>	No	The default value that the property takes. It can be empty.	String
<code>description</code>	No	The description of the property. It can be empty.	String
<code>encoded</code>	No	Specifies whether to encode the value of this property.	Boolean

The following example shows a sample `properties` field:

```
properties:
 ID:
 value: 1234
 description: An ID to be used for validating.
 encoded: false
```

## catalogs

Use the `catalogs` extension to assign catalog-specific values to properties defined in the `properties` section.

The `catalogs` section has the following structure:

```
catalogs:
 catalog_name_1:
 properties:
 property_name_1: value_1
 property_name_2: value_2
 catalog_name_2:
 properties:
 property_name_1: value_3
```

The following table lists the fields found in the `catalogs` extension:

Table 1. Catalogs properties

Property	Required	Description	Data type
<code>catalog_name</code>	Yes	The name of the catalog. It must match a catalog in API Manager. Note that this is the field name, not the contents of the field.	Object
<code>properties</code>	Yes	This field contains any properties that are to be set.	Object
<code>property_name</code>	Yes	The field name is the property to be set and must match a property defined in the <code>properties</code> extension. It contains the value of the property for the catalog to which the field belongs. If, in the <code>properties</code> extension, it has been specified as encoded, this will be encoded when saved or staged by IBM® API Connect.	String

The following is an example of a `catalogs` extension:

```
catalogs:
 Sandbox:
 properties:
 ID: 5678
```



## activity-log

If you're using the DataPower® API Gateway, you can use the `activity-log` extension to configure your logging preferences for the API activity that is stored in Analytics. The preferences that you specify will override the default settings for collecting and storing details of the API activity.

An API event record exists for each API execution event in the Gateway server. By default, the content type that is collected and stored in API event records is `activity` for when API execution completes successfully, and `payload` for when API execution completes with an error code. When you compose your API definition, you can change the type of content to log in these API event records. For more information about API event records, see [API event record fields](#).

Note that if you're using the DataPower Gateway (v5 compatible), you can configure your logging preferences by using an `activity-log` policy in your API assembly. For more information, see [activity-log](#).

The `activity-log` extension takes the following form:

```
activity-log:
 success-content: activity_to_log_if_call_successful
 error-content: activity_to_log_if_call_unsuccessful
 enabled: is_activity_logging_enabled
```

Note: If payload logging is enabled, for the gateway to capture payloads buffering must also be enabled.

```
activity-log:
 success-content: activity_to_log_if_call_successful
 error-content: activity_to_log_if_call_unsuccessful
 enabled: is_activity_logging_enabled
 buffering: true
```

The following table shows the properties of the `activity-log` extension:

Table 1. The properties of the activity-log extension

Property	Required	Description	Type
<code>success-content</code>	No	Defines the type of content to be logged when the operation is successful. Valid values: <ul style="list-style-type: none"> <li><code>none</code>: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal.</li> <li><code>activity</code>: Logs invocation only (only the resource URI is recorded).</li> <li><code>header</code>: Logs activity and header.</li> <li><code>payload</code>: Logs activity, header, and payload (the original request, if any, and the final response).</li> </ul> The default value is <code>activity</code> .	String
<code>error-content</code>	No	Indicates what content to log when an error occurs. Valid values: <ul style="list-style-type: none"> <li><code>none</code>: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal.</li> <li><code>activity</code>: Logs invocation only (only the resource URI is recorded).</li> <li><code>header</code>: Logs activity and header.</li> <li><code>payload</code>: Logs activity, header, and payload (the original request, if any, and the final response).</li> </ul> The default value is <code>payload</code> .	String
<code>enabled</code>	No	Indicates whether activity logging is enabled or disabled.	Boolean

## Example 1

---

Override the default activity logging settings:

```
activity-log:
 success-content: none
 error-content: header
 enabled: true
```

## Example 2

---

Turn off activity logging:

```
activity-log:
 enabled: false
```

---

## Specifying a gateway type for an API definition

An API definition is specific to one or other of the gateway types, DataPower® API Gateway or DataPower Gateway (v5 compatible). API definitions must specify which type of gateway the API uses.

### Before you begin

---

IBM® API Connect supports two gateway types: DataPower Gateway (v5 compatible) and DataPower API Gateway.

DataPower Gateway (v5 compatible) has been available with IBM API Connect for a number of years. The DataPower API Gateway is a new gateway that has been designed with APIs in mind, and with the same security focus as DataPower Gateway (v5 compatible).

For more information on how to choose which gateway type to use, see [API Connect gateway types](#).

### About this task

---

You must specify which type of gateway each API uses. APIs can use only one type of gateway.

When you modify your API definitions to use a specific gateway type, you must ensure that each policy and policy version in the API are supported by the gateway type. DataPower Gateway (v5 compatible) and DataPower API Gateway each support policies that the other gateway type does not. In some cases, the same policy is supported by both gateway types, but with a different version number.

For example, DataPower Gateway (v5 compatible) supports version **1.0.0** of the **invoke** policy, but DataPower API Gateway requires version **2.0.0**.

For information on policies, see [execute](#).

For information on policy versions, see the documentation for each individual policy. For example, to review the **invoke** policy, see [invoke](#).

### Procedure

---

1. Edit the API definition YAML file and add the gateway type. For example:

- DataPower Gateway (v5 compatible):

```
gateway: datapower-gateway
```

- DataPower API Gateway

```
gateway: datapower-api-gateway
```

Note: If your API YAML file includes the **gateway**: property, the **enforced**: property must be set to **true**

2. Ensure that the policies, including policy versions, in your APIs are supported by the gateway type. Review the list of policies on the [execute](#) page. If necessary, review the individual policy pages to determine the supported version number.
3. Ensure that any Product that uses this API is configured to use the same gateway type. See [Specifying a gateway type for your Product](#).

### Related information

---

- [DataPower API Gateway porting notes](#)

---

## Referring to an extension in an API definition

To refer to an OpenAPI extension in your API definition YAML (or JSON) file, add an **extensions** key under **x-ibm-configuration**.

### Before you begin

---

Add the OpenAPI extension to any Catalogs that the API is to be published to, by importing its definition file; for details, see [Working with OpenAPI extensions](#). If the target Catalog does not contain the OpenAPI extension then attempts to publish the API to it will fail.

## About this task

The following example shows an extension definition file that defines a bank branch, in YAML format:

```
extension: '1.0.0'

info:
 title: Banking services
 name: banking
 version: 1.0.0
 description: Banking extensions
 contact:
 name: IBM API Connect
 url: https://apiconnect.ibm.com/
 email: myname@ibm.com

portal-visible: true

properties:
 title: "Branch"
 type: "object"
 properties:
 Branch type:
 type: "string"
 enum:
 - "ATM"
 - "Walk in"
 location:
 type: "object"
 title: "Location"
 properties:
 city:
 type: "string"
 default: "San Francisco"
 state:
 type: "string"
 default: "CA"
 citystate:
 type: "string"
 description: "This is generated automatically from the previous two fields"
 template: "{{city}}, {{state}}"
 watch:
 city: "location.city"
 state: "location.state"
 required:
 - Branch type
```

## Procedure

1. Edit the API definition YAML file and add the reference. For example:

```
x-ibm-configuration:
 .
 .
 .
 extensions:
 banking: 1.0.0
```

2. In your API definition file, add a property whose name must begin with `x-` and references the name of the extension you created. Specify the values of the properties that were defined in the OpenAPI extension YAML file. The following example uses the banking extension:

```
.
.
.
x-banking:
 Branch type: ATM
 location:
 city: San Francisco
 state: CA
 citystate: 'San Francisco, CA'
```

## Using \$ref to reuse code fragments in your OpenAPI files

If you deploy an API to an IBM® API Connect Management server by using the developer toolkit command line, you can use the `$ref` field in your OpenAPI YAML and JSON API definition files to reference a fragment of OpenAPI code that is defined in a separate file. When API Connect processes the source API definition file, the `$ref` field is replaced with the contents of the target file.

Use the following syntax in your source YAML file:

```
$ref: path_to_file_containing_code_fragment
```

Use the following syntax in your source JSON file:

```
{
 $ref: path_to_file_containing_code_fragment
}
```

```
}
```

For example:

```
$ref: code_fragments/my_fragment.yaml

{
 "$ref": "code_fragments/my_fragment.json"
}
```

The replacement of the `$ref` field with the target code fragment occurs when you perform any of the following actions on the API defined by the source API definition file:

- Stage or publish an API to an API Connect Management server by using the **apic publish** command. For more information, see [Publishing APIs and applications](#).
- Validate the API definition YAML file by using the **apic validate** command. For more information, see [Validating the YAML or JSON definition of an API or Product](#).

Important: You cannot insert a `$ref` field at the root level of your OpenAPI file.

## Example 1

---

A source YAML file contains the following OpenAPI code:

```
swagger: '2.0'
info:
 version: 1.0.0
 title: Branches
 x-ibm-name: Branches
 description: Provides operations relating to BankA branch information.
basePath: /branches
paths:
 /details:
 get:
 responses:
 - $ref: code_fragments/paths.yaml#/details/get/responses
 .
 .
 .
```

The file `paths.yaml` contains the following OpenAPI code fragment:

```
details:
 get:
 responses:
 '200':
 description: 200 OK defined in $ref file
 schema:
 $ref: '#/definitions/branch'
 summary: Branch details
 description: Retrieve details of the current branches of BankA.
```

When API Connect processes the source YAML file, the `$ref` field is replaced with the target code fragment, yielding the following OpenAPI code:

```
swagger: '2.0'
info:
 version: 1.0.0
 title: Branches
 x-ibm-name: Branches
 description: Provides operations relating to BankA branch information.
basePath: /branches
paths:
 /details:
 get:
 responses:
 '200':
 description: 200 OK
 schema:
 $ref: '#/definitions/branch'
 summary: Branch details
 description: Retrieve details of the current branches of BankA.
 .
 .
 .
```

---

## Creating a Product definition file

You define a Product by creating a Product definition file.

### About this task

---

You can create a Product definition file in either of the following ways:

- Create the file in an editor of your choice.
- Create a Product definition file by using the **apic create:product** command and then modify it. You can base your Product definition file on the default Product template or on your own custom template.

This topic describes the syntax that is needed for both options and also the structure needed when the file is created in the editor. For details on what you can include in the product definition file, see [Product definition schema](#).

You can create a Product in the CLI by running `apic create:product` and supplying additional arguments on the command line. For example, you can enter the following text in a single command:

```
apic create:product --name product_name --title product_title --filename product_file_name.yaml
--apis "filename_of_api1.yaml filename_of_api2.yaml"
```

where:

- `product_name` is the name for your new Product.
- `product_title` is the title of your new Product.
- `product_file_name.yaml` is the name of the yaml file that is created for your new Product.
- `filename_of_api1.yaml` is the file name of one of the APIs that is used within the new Product.
- `filename_of_api2.yaml` is the file name of one of the APIs that is used within the new Product.

Another option is to create a Product interactively in the command line by running `apic create:product` and following the prompts.

You can see further details and available options for the `apic create:product` command by entering the command:

```
apic create:product --help
```

You can also create a Product from a custom Handlebars template by using the following command:

```
apic create:product --template template_filename --title product_title
```

where `template_filename` is the name of the Handlebars template to use, and `product_title` is the title of your Product. A Product template file must have a `.hbs` file name extension. You can create a template from scratch, or start with the example (default) Product template provided in [API and Product definition template examples](#).

A Product definition file contains the following sections:

- The specification version
- An information section
- A visibility section
- An APIs section
- A Plans section

In this topic, YAML is used, but the instructions can be adapted to use JSON. Both `.yaml` and `.yaml` file extensions are supported, but the use of the `.yaml` file extension is recommended by [yaml.org](#).

Note: All keys and enumeration values that are specified in this topic are case-sensitive.

## Procedure

---

Structure your Product definition file by completing the following steps:

1. Set the specification version by adding the following line to the beginning of the file: `product: 1.0.0`.  
This line specifies the template version, and is always `product: 1.0.0`.  
Note: The specification version is distinct from your Product version. The specification version refers to this YAML file, while the Product version is by your discretion.
2. Include an information section with details about the Product, as described in [Completing the information section of your Product description](#).
3. Include a visibility section that specifies who can view and subscribe to the Product, as described in [Specifying the visibility of your Product](#).
4. Include an APIs section that references the APIs to be included in the Product, as described in [Referencing the APIs for your Product](#).
5. Include a Plans section that described the Plans you want include in your Product, as described in [Describing Plans in your Product](#).

## Results

---

You have completed a YAML representation of your Product. A complete example with full indentation can be found in [An example YAML representation of a Product](#).

You can create multilingual API and Product documentation by using an `x-ibm-languages` extension directly in the OpenAPI definition. For more information, see [Using x-ibm-languages to create multilingual API and Product documentation](#).

- [Product definition schema](#)  
The product definition schema defines the properties (and their data types) that you can include in a product definition file.
- [Converting a Product YAML file to use DataPower API Gateway](#)  
IBM API Connect now supports DataPower® API Gateway. You can optionally specify this gateway type in your Product YAML file.

## Related information

---

- [Changing the availability of a Product](#)

## Product definition schema

The product definition schema defines the properties (and their data types) that you can include in a product definition file.

```
{
 "id": "https://api.ibm.com/product/schema.json#",
 "$schema": "http://json-schema.org/draft-06/schema#",
 "title": "A schema to represent the IBM specific API configuration",
 "type": "object",
 "additionalProperties": false,
 "required": [
 "product",
 "info"
]
}
```



```

},
"patternProperties": {
 "^x-": {}
},
"properties": {
 "product": {
 "type": "string",
 "enum": [
 "1.0.0"
]
 },
 "info": {
 "$ref": "#/definitions/InfoSection"
 },
 "visibility": {
 "type": "object",
 "additionalProperties": false,
 "required": [
 "view",
 "subscribe"
],
 "patternProperties": {
 "^x-": {}
 },
 "properties": {
 "view": {
 "type": "object",
 "additionalProperties": false,
 "required": [
 "type"
],
 "properties": {
 "type": {
 "enum": [
 "public",
 "authenticated",
 "custom"
]
 },
 "orgs": {
 "type": "array",
 "items": {
 "type": "string"
 }
 },
 "tags": {
 "type": "array",
 "items": {
 "type": "string"
 }
 },
 "enabled": {
 "type": "boolean"
 }
 }
 },
 "subscribe": {
 "type": "object",
 "additionalProperties": false,
 "required": [
 "type"
],
 "properties": {
 "type": {
 "enum": [
 "authenticated",
 "custom"
]
 },
 "orgs": {
 "type": "array",
 "items": {
 "type": "string"
 }
 },
 "tags": {
 "type": "array",
 "items": {
 "type": "string"
 }
 },
 "enabled": {
 "type": "boolean"
 }
 }
 }
 }
 },
 "properties": {
 "type": "object",
 "propertyNames": {
 "pattern": "^[^]|([^\].*[^\])$"
 },
 "additionalProperties": {
 "type": "object",
 "additionalProperties": false,

```

```

 "properties": {
 "value": {
 "type": "string"
 },
 "description": {
 "type": "string"
 },
 "encoded": {
 "type": "boolean"
 }
 }
 },
 "catalogs": {
 "type": "object",
 "propertyNames": {
 "pattern": "^[^]|([^].*[^])$"
 },
 "additionalProperties": {
 "type": "object",
 "additionalProperties": false,
 "properties": {
 "properties": {
 "type": "object",
 "additionalProperties": {
 "type": "string"
 }
 }
 }
 }
 },
 "gateways": {
 "type": "array",
 "minItems": 1,
 "items": {
 "enum": [
 "micro-gateway",
 "datapower-gateway",
 "datapower-api-gateway",
 "event-gateway"
]
 }
 },
 "apis": {
 "type": "object",
 "propertyNames": {
 "pattern": "^[^]|([^].*[^])$"
 },
 "additionalProperties": {
 "type": "object",
 "additionalProperties": false,
 "properties": {
 "$ref": {
 "type": "string"
 },
 "name": {
 "type": "string"
 }
 }
 }
 },
 "oauthProviders": {
 "type": "array",
 "propertyNames": {
 "pattern": "^[^]|([^].*[^])$"
 },
 "items": {
 "type": "string"
 }
 },
 "billings": {
 "type": "object",
 "propertyNames": {
 "pattern": "^[^]|([^].*[^])$"
 },
 "additionalProperties": {
 "type": "object",
 "additionalProperties": false,
 "properties": {
 "name": {
 "type": "string"
 }
 }
 }
 },
 "plans": {
 "type": "object",
 "propertyNames": {
 "pattern": "^[^]|([^].*[^])$"
 },
 "additionalProperties": {
 "type": "object",
 "additionalProperties": false,
 "patternProperties": {
 "^x-": {}
 }
 }
 },

```

```

"required": [
 "title"
],
"properties": {
 "title": {
 "type": "string"
 },
 "description": {
 "type": "string"
 },
 "approval": {
 "type": "boolean"
 },
 "rate-limit": {
 "$ref": "#/definitions/RateLimitSection"
 },
 "rate-limits": {
 "$ref": "#/definitions/RateLimitsSection"
 },
 "burst-limits": {
 "$ref": "#/definitions/BurstLimitsSection"
 },
 "assembly-rate-limits": {
 "$ref": "#/definitions/AssemblyRateLimitsSection"
 },
 "assembly-burst-limits": {
 "$ref": "#/definitions/AssemblyBurstLimitsSection"
 },
 "assembly-count-limits": {
 "$ref": "#/definitions/AssemblyCountLimitsSection"
 },
 "billing": {
 "$ref": "#/definitions/BillingSection"
 },
 "graphql": {
 "$ref": "#/definitions/GraphQLSection"
 },
 "apis": {
 "type": "object",
 "propertyNames": {
 "pattern": "^[^]|([]).*[^]$"
 },
 "additionalProperties": {
 "type": "object",
 "additionalProperties": false,
 "properties": {
 "operations": {
 "type": "array",
 "items": {
 "type": "object",
 "additionalProperties": false,
 "properties": {
 "operationId": {
 "type": "string"
 },
 "path": {
 "type": "string"
 },
 "operation": {
 "type": "string"
 },
 "rate-limit": {
 "$ref": "#/definitions/RateLimitSection"
 },
 "rate-limits": {
 "$ref": "#/definitions/RateLimitsSection"
 }
 }
 }
 },
 "graphql-schema-options": {
 "$ref": "#/definitions/GraphQLSchemaOptions"
 }
 }
 }
 }
},
"definitions": {
 "GraphQLSchemaOptions": {
 "type": "object",
 "additionalProperties": false,
 "required": [
 "visibility-list"
],
 "properties": {
 "visibility-list": {
 "type": "object",
 "additionalProperties": false,
 "required": [
 "type",
 "values"
],
 "properties": {

```

```

 "type": {
 "type": "string",
 "enum": [
 "remove"
]
 },
 "values": {
 "type": "array",
 "items": {
 "$ref": "#/definitions/NonEmptyString"
 }
 }
 }
 },
 },
 "InfoSection": {
 "type": "object",
 "additionalProperties": false,
 "required": [
 "title",
 "version"
],
 "patternProperties": {
 "^x-": {}
 },
 "properties": {
 "name": {
 "type": "string",
 "pattern": "^[a-zA-Z0-9._]+(-)*([a-zA-Z0-9])$"
 },
 "version": {
 "type": "string"
 },
 "title": {
 "type": "string"
 },
 "description": {
 "type": "string"
 },
 "termsOfService": {
 "type": "string"
 },
 "contact": {
 "type": "object",
 "additionalProperties": false,
 "patternProperties": {
 "^x-": {}
 },
 "properties": {
 "name": {
 "type": "string"
 },
 "url": {
 "type": "string",
 "format": "uri"
 },
 "email": {
 "type": "string",
 "format": "email"
 }
 }
 },
 "license": {
 "type": "object",
 "additionalProperties": false,
 "patternProperties": {
 "^x-": {}
 },
 "required": [
 "name"
],
 "properties": {
 "name": {
 "type": "string"
 },
 "url": {
 "type": "string",
 "format": "uri"
 }
 }
 },
 "summary": {
 "type": "string",
 "description": "Short description of the API."
 },
 "categories": {
 "type": "array",
 "items": {
 "type": "string"
 }
 }
 }
 },
 "RateLimitSection": {
 "type": "object",

```

```

"additionalProperties": false,
"required": [
 "value"
],
"patternProperties": {
 "^x-": {}
},
"properties": {
 "value": {
 "$ref": "#/definitions/RateLimitValue"
 },
 "hard-limit": {
 "$ref": "#/definitions/RateLimitHardLimit"
 }
}
},
"AssemblyRateLimitSection": {
 "type": "array",
 "minItems": 1,
 "items": {
 "type": "object",
 "additionalProperties": false,
 "required": [
 "value",
 "hard-limit",
 "cache-only",
 "is-client",
 "use-api-name",
 "use-app-id",
 "use-client-id"
],
 "patternProperties": {
 "^x-": {}
 },
 "properties": {
 "value": {
 "$ref": "#/definitions/RateLimitValue"
 },
 "hard-limit": {
 "$ref": "#/definitions/RateLimitHardLimit"
 },
 "cache-only": {
 "type": "boolean"
 },
 "is-client": {
 "type": "boolean"
 },
 "use-api-name": {
 "type": "boolean"
 },
 "use-app-id": {
 "type": "boolean"
 },
 "use-client-id": {
 "type": "boolean"
 },
 "dynamic-value": {
 "type": "string"
 },
 "weight": {
 "$ref": "#/definitions/NonEmptyString"
 }
 }
 }
},
"AssemblyBurstLimitSection": {
 "type": "array",
 "minItems": 1,
 "items": {
 "type": "object",
 "additionalProperties": false,
 "required": [
 "value",
 "cache-only",
 "is-client",
 "use-api-name",
 "use-app-id",
 "use-client-id"
],
 "patternProperties": {
 "^x-": {}
 },
 "properties": {
 "value": {
 "$ref": "#/definitions/RateLimitValue"
 },
 "cache-only": {
 "type": "boolean"
 },
 "is-client": {
 "type": "boolean"
 },
 "use-api-name": {
 "type": "boolean"
 },
 "use-app-id": {

```

```

 "type": "boolean"
 },
 "use-client-id": {
 "type": "boolean"
 },
 "dynamic-value": {
 "type": "string"
 },
 "weight": {
 "$ref": "#/definitions/NonEmptyString"
 }
}
},
"NonEmptyString": {
 "type": "string",
 "pattern": ".+"
},
"AssemblyCountLimitSection": {
 "type": "array",
 "minItems": 1,
 "maxItems": 1,
 "items": {
 "type": "object",
 "additionalProperties": false,
 "required": [
 "value",
 "hard-limit",
 "cache-only",
 "is-client",
 "use-api-name",
 "use-app-id",
 "use-client-id"
],
 "patternProperties": {
 "^x-": {}
 },
 "properties": {
 "value": {
 "oneOf": [
 {
 "type": "integer"
 },
 {
 "type": "string",
 "enum": [
 "unlimited"
]
 }
]
 },
 "hard-limit": {
 "$ref": "#/definitions/RateLimitHardLimit"
 },
 "cache-only": {
 "type": "boolean"
 },
 "is-client": {
 "type": "boolean"
 },
 "use-api-name": {
 "type": "boolean"
 },
 "use-app-id": {
 "type": "boolean"
 },
 "use-client-id": {
 "type": "boolean"
 },
 "dynamic-value": {
 "type": "string"
 },
 "weight": {
 "$ref": "#/definitions/NonEmptyString"
 },
 "auto-decrement": {
 "type": "boolean"
 }
 }
 }
},
"RateLimitValue": {
 "type": "string",
 "pattern": "^(?([0-9]|[1-9][0-9]{1,8}|[1-3][0-9]{9}|4[01][0-9]{8}|42[0-8][0-9]{7}|429[0-3][0-9]{6}|4294[0-8][0-9]{5}|42949[0-5][0-9]{4}|429496[0-6][0-9]{3}|4294967[01][0-9]{2}|42949672[0-8][0-9]|429496729[0-5])\\/(?[0-9]|[1-9][0-9]{1,3}|[1-5][0-9]{4}|6[0-4][0-9]{3}|65[0-4][0-9]{2}|655[0-2][0-9]|6553[0-5])?(second|minute|hour|day|week)|unlimited)$"
},
"RateLimitHardLimit": {
 "type": "boolean",
 "default": false
},
"RateLimitsSection": {
 "type": "object",
 "patternProperties": {
 "^x-": {}
 }
},

```

```

 "additionalProperties": {
 "type": "object",
 "oneOf": [
 {
 "$ref": "#/definitions/RateLimitSection"
 }
]
 },
 },
 "AssemblyRateLimitsSection": {
 "type": "object",
 "patternProperties": {
 "^x-": {}
 },
 },
 "additionalProperties": {
 "type": "object",
 "$ref": "#/definitions/AssemblyRateLimitSection"
 }
},
"AssemblyBurstLimitsSection": {
 "type": "object",
 "patternProperties": {
 "^x-": {}
 },
 "additionalProperties": {
 "type": "object",
 "$ref": "#/definitions/AssemblyBurstLimitSection"
 }
},
"AssemblyCountLimitsSection": {
 "type": "object",
 "patternProperties": {
 "^x-": {}
 },
 "additionalProperties": {
 "type": "object",
 "$ref": "#/definitions/AssemblyCountLimitSection"
 }
},
"BurstLimitsSection": {
 "type": "object",
 "additionalProperties": {
 "type": "object",
 "additionalProperties": false,
 "required": [
 "value"
],
 "patternProperties": {
 "^x-": {}
 },
 },
 "properties": {
 "value": {
 "type": "string",
 "pattern": "^(\\d+\\\\\\\\(\\d+)?(second|minute))$"
 }
 }
},
},
"GraphQLSection": {
 "type": "object",
 "additionalProperties": false,
 "properties": {
 "per-query-policy": {
 "type": "object",
 "additionalProperties": false,
 "properties": {
 "maximum-nesting": {
 "type": "integer"
 },
 "maximum-resolve-complexity": {
 "type": "integer"
 },
 "maximum-type-complexity": {
 "type": "integer"
 }
 }
 },
 "rate-limits": {
 "type": "object",
 "additionalProperties": {
 "type": "object",
 "oneOf": [
 {
 "$ref": "#/definitions/GraphQLRateLimitSection"
 }
]
 }
 }
 }
},
},
"GraphQLRateLimitSection": {
 "type": "object",
 "additionalProperties": false,
 "required": [
 "resolve-complexity",

```

```

 "type-complexity"
],
 "properties": {
 "resolve-complexity": {
 "$ref": "#/definitions/RateLimitSection"
 },
 "type-complexity": {
 "$ref": "#/definitions/RateLimitSection"
 }
 }
},
"BillingSection": {
 "type": "object",
 "additionalProperties": false,
 "properties": {
 "billing": {
 "type": "string"
 },
 "currency": {
 "type": "string"
 },
 "price": {
 "type": "string",
 "pattern": "^\\d+(\\.\\d+)?$"
 },
 "period": {
 "type": "integer"
 },
 "period-unit": {
 "type": "string",
 "enum": [
 "month"
]
 },
 "trial-period": {
 "type": "integer"
 },
 "trial-period-unit": {
 "type": "string",
 "enum": [
 "day"
]
 }
 }
}
}
}

```

## Completing the information section of your Product description

Provide users with information about your Product.

### About this task

The information section of a YAML representation of a Product contains the Product's name and version. It can also include contact and license details.

Note: All keys and enumeration values that are specified in this topic are case-sensitive.

An example information section for a YAML representation of a Product can be found at the end of this topic. An example of a complete YAML representation of a Product can be found in [An example YAML representation of a Product](#).

### Procedure

To complete the information section of your Product description, complete the following steps.

1. Begin the section and provide a name, title, description, and version number; use the following syntax:

```

info:
 version: Product_Version
 title: Product_Title
 name: Short_Name
 summary: Product_Description

```

where:

- *version* is the version number of the policy.  
Tip: The **version.release.modification** version numbering scheme is recommended, for example `1.0.0`.
  - *Product\_Title* is the title of the Product. Any string can be used, but the title should be kept short so that it can be displayed in the API Manager user interface.
  - *Short\_Name* is the short name for the policy. It must be a single word and contain only alphanumeric characters, and the - (dash) and \_ (underscore) characters. The name is case-sensitive, and should be 20 characters or fewer so that it can be displayed in the API Manager user interface of IBM API Connect.
  - *Product\_Description* is a short description of the policy. Any string can be used.
2. Optional: Under info, supply contact information; use the following syntax:

```

contact:
 name: Contact_Name

```



```
url: 'Contact_URL'
email: Contact_email
```

where:

- *Contact\_Name* is the name of the contact for this Product.
- *Contact\_URL* is the URL for contact your organization.
- *Contact\_email* is an email for contacting your organization.

3. Optional: Under info, add license information for your Product; use the following syntax:

```
license:
 name: License_Name
 url: 'License_URL'
```

where:

- *License\_Name* is the name of the license that applied to your Product.
- *License\_URL* is the URL at which information about the license can be accessed.

4. Optional: Under info, add your terms of service; use the following syntax:

```
termsOfService: Terms
```

where *Terms* is a string that details the terms of service for using your Product.

## Results

---

You have completed the information section of your Product's YAML representation. It should have the following form:

```
info:
 version: Product_Version
 title: Product_Title
 name: Short_Name
 description: Product_Description
 contact:
 name: Contact_Name
 url: 'Contact_URL'
 email: Contact_email
 license:
 name: License_Name
 url: 'License_URL'
 termsOfService: Service_Terms
```

where all variables are as described in previously in this topic. The indentation must be as in the example.

---

## Specifying a gateway type for your Product

Product definitions must specify which type of gateway the Product uses.

### Before you begin

---

IBM® API Connect supports two gateway types: DataPower® Gateway (v5 compatible) and DataPower API Gateway.

DataPower Gateway (v5 compatible) has been available with IBM API Connect for a number of years. The DataPower API Gateway is a new gateway that has been designed with APIs in mind, and with the same security focus as DataPower Gateway (v5 compatible).

For more information on how to choose which gateway type to use, see [API Connect gateway types](#).

### About this task

---

You can specify the gateway type for a Product.

Products can use only one type of gateway, and the APIs in the Product must use the same type of gateway. See [Specifying a gateway type for an API definition](#).

### Procedure

---

1. Determine the gateway type that is configured for the APIs in your Product. The Product gateway type must match the APIs gateway type. To review the APIs for your Product, see [Referencing the APIs for your Product](#).

2. Edit the Product definition file (YAML source) to add the gateway type you want to use:

- DataPower Gateway (v5 compatible):

```
gateways:
 datapower-gateway
```

- DataPower API Gateway

```
gateways:
 datapower-api-gateway
```

3. Ensure that the policies, including policy versions, in the APIs in your Product are supported by the gateway type. Review the list of policies on the [execute](#) page. If necessary, review the individual policy pages to determine the supported version number.

---

## Related information

- [DataPower API Gateway porting notes](#)

---

## Converting a Product YAML file to use DataPower API Gateway

IBM® API Connect now supports DataPower® API Gateway. You can optionally specify this gateway type in your Product YAML file.

---

### Before you begin

IBM API Connect provides two gateway types, DataPower Gateway (v5 compatible) and DataPower API Gateway.

DataPower Gateway (v5 compatible) has been available with IBM API Connect for a number of years. The DataPower API Gateway is a new gateway that has been designed with APIs in mind, and with the same security focus as DataPower Gateway (v5 compatible).

For more information on how to choose which gateway type to use, see [API Connect gateway types](#).

---

### About this task

Product definitions can specify only one type of gateway. The type of gateway must match the gateway type that is used by the APIs that are included in the product. If you want to take advantage of the capabilities of DataPower API Gateway, you can easily modify your existing Product definitions to specify it as the gateway type.

---

### Procedure

1. Verify that the APIs in the product specify DataPower API Gateway as the gateway type, and that the APIs have been converted for use with the DataPower API Gateway.
2. Open the Product YAML file for editing.  
For example:

```
product: '1.0.0'
info:
 name: testprod
 title: testprod
 version: 1.0.0
 .
 .
 .
```

3. Add a **gateways**: setting, and specify **datapower-api-gateway**.  
For example:

```
product: '1.0.0'
info:
 name: testprod
 title: testprod
 version: 1.0.0
gateways:
 - datapower-api-gateway
 .
 .
 .
```

4. Save the modified YAML file.

---

## Specifying the visibility of your Product

Detail who can view and subscribe to your Product.

---

### About this task

The visibility section of your Product's YAML representation determines who can view and subscribe to your Product in IBM® API Connect.

Note: All keys and enumeration values that are specified in this topic are case-sensitive.

An example visibility section of a YAML representation of a Product can be found at the end of this topic. An example of a complete YAML representation of a Product can be found in [An example YAML representation of a Product](#).

---

### Procedure

To complete the visibility section of your Product description, complete the following steps:

1. Title the section by adding **visibility**:
2. Under visibility, specify who can view the Product; use the following syntax:

```
view:
 enabled: View_Toggle
```

```
type: View_Audience
tags:
- View_Tag_1
- View_Tag_2
orgs: View_Organizations
```

where

- *View\_Toggle* determines whether the Product is visible to anybody or not. It must be `true` or `false`. If `false`, the Product will not be visible in the Developer Portal.
- *View\_Audience* must be `public`, in which case the Product is visible to anybody who uses the Developer Portal, `authenticated`, in which case the Product is visible to anybody registered through the Developer Portal, or `custom`, in which case the Product is visible to a specified group of users.
- The *View\_Tag* variables are strings that contains any tags that you want to attach to your Product's view status. Each tag must be on a new line and preceded with a dash. If you do not want to use any tags, do not include `tags:`.
- *View\_Organizations* specifies the organizations that can view the Product if *View\_Audience* is set to `custom`. If *View\_Audience* is not set to `custom`, do not include `orgs:`. If *View\_Audience* is `custom`, specify organizations in the following manner:

```
orgs:
- Organization_1
- Organization_2
```

where the *Organization* variables are the names of the organizations that are to be allowed to view the Product.

3. Under visibility, specify who can subscribe to your Product's Plans; use the following syntax:

```
subscribe:
enabled: Subscribe_Toggle
type: Subscribe_Audience
tags:
- Subscribe_Tag_1
- Subscribe_Tag_2
orgs: Subscribe_Organizations
```

where:

- *Subscribe\_Toggle* must be `true` or `false`. If `false`, no developers will be able to subscribe to the Product.
- *Subscribe\_Audience* must be `authenticated`, in which case the Product's Plans can be subscribed to by anybody who is registered through the Developer Portal, or `custom`, in which case the Product's Plans can be subscribed to by a specified group of users.
- The *Subscribe\_Tag* variables are strings that contains any tags you want to attach to your Product. Each tag must be on a new line and preceded by a dash.
- *Subscribe\_Organizations* specifies the organizations that can subscribe to the Product's Plans if *Subscribe\_Audience* is set to `custom`. If *Subscribe\_Audience* is not set to `custom`, omit it or set as `[]`. If *Subscribe\_Audience* is `custom`, specify organizations in the following manner:

```
orgs:
- Organization_1
- Organization_2
```

where the *Organization* variables are the names of the organizations that are to be allowed to subscribe to the Product.

Note: If a Product is to be available for subscription, it must already be visible. As a result, for the subscription audience to be `authenticated`, the view audience cannot be `custom`.

## Results

---

You have completed the visibility section of your Product's YAML representation. It should have the following form:

```
visibility:
 view:
 enabled: View_Toggle
 type: View_Audience
 tags:
 - View_Tag_1
 - View_Tag_2
 orgs:
 - Organization_1
 - Organization_2

 subscribe:
 enabled: Subscribe_Toggle
 type: Subscribe_Audience
 tags:
 - Subscribe_Tag_1
 - Subscribe_Tag_2
 orgs:
 - Organization_1
 - Organization_2
```

where all variables are as described in previously in this topic. The indentation must be as in the example.

## What to do next

---

You can also define the visibility of a Product at a Catalog or Space level. For more information, see [Configuring Product visibility in a Catalog](#).

## Related information

---

- [Configuring Product visibility in a Catalog](#)

## Referencing the APIs for your Product

Detail the file paths for the APIs you want to include in your Product in IBM® API Connect.

## About this task

---

Before an API can be included in a Plan, it must first be referenced in the APIs section of your Product description.

## Procedure

---

Begin the APIs section and reference the APIs you want to include in your Product. The syntax that you use depends on which user interface you are using, as follows:

- API Designer:

```
apis:
 API_1_NameAPI_1_Version
 $ref: API_1_File_Path
 API_2_NameAPI_2_Version
 $ref: API_2_File_Path
```

- API Manager:

```
apis:
 API_1_NameAPI_1_Version
 name: 'API_1_Name:API_1_Version'
 API_2_NameAPI_2_Version
 name: 'API_2_Name:API_2_Version'
```

where

- the *API\_n\_Name* variables are the case-sensitive names of your APIs.
- the *API\_n\_Version* variables are the versions of your APIs.
- the *API\_n\_File\_Path* variables are the file paths for the YAML files containing OpenAPI representations of your APIs.

The following example shows a complete sample APIs section:

```
apis:
 api11.0.0:
 name: 'api1:1.0.0'
 api21.0.0:
 name: 'api2:1.0.0'
 api31.0.0:
 name: 'api3:1.0.0'
```

For more information about creating OpenAPI definitions, see [Creating an OpenAPI definition file](#).

The indentation must be as in the examples and the APIs must have been created before they can be referenced.

## Results

---

You have referenced the APIs that are to be included in your Product. An example of a complete YAML representation of a Product can be found in [An example YAML representation of a Product](#).

## Describing Plans in your Product

---

Describe the Plans that you want to include in your IBM® API Connect Product and which APIs they will contain, as well as any rate limits that apply.

## About this task

---

A Plan contains APIs and their operations. It can be used to implement rate limits and tailor visibility.

Note: If you are using more than one DataPower® server in a Gateway service, then to properly calculate API calls for rate limits the servers must be able to communicate with each other by using SLM peer groups, using either SLM unicast peering or SLM multicast peering depending on your network configuration. For more information, see [SLM peering](#).

In addition, you can apply burst limits to your Plans, to prevent usage spikes that might damage infrastructure. Multiple burst limits can be set per Plan, at second and minute time intervals. You can also set multiple rate limits per Plan and per operation, at second, minute, hour, day, and week time intervals.

Note: All keys and enumeration values that are specified in this topic are case-sensitive.

Two example Plan descriptions from a YAML representation of a Product can be found at the end of this topic. An example of a complete YAML representation of a Product can be found in [An example YAML representation of a Product](#).

## Procedure

---

To describe your Plans, include APIs in them, and set rate limits for Plans or specific operations, complete the following instructions:

1. Begin the Plans section with `plans`:
2. Under Plans, begin the description of your first Plan by providing a name, description, and specifying whether approval is required for subscription requests; use the following syntax:

```
plans:
 Plan_Name:
 title: Plan Title
 description: Plan Description
 approval: Approval_Toggle
```

where:

- *Plan\_Name* is the name of the Plan. It must be a single word and contain only alphanumeric characters, and the - (dash) and \_ (underscore) characters. The name is case-sensitive, and should be 20 characters or fewer so that it can be displayed in the API Manager user interface.
- *Plan\_Title* is the title of the Plan. Any string can be used, but the title should be kept short so that it can be displayed in the API Manager user interface.
- *Plan\_Description* is a description of your Plan.
- *Approval\_Toggle* must be either `true`, in which case approval is needed for subscription requests, or `false`, in which case subscriptions are automatically approved.

3. Optional: Under your Plan, add multiple burst and rate limits that will be shared across all the operations in the Plan; use the following syntax:

Note: For information about rate limits and burst limits in API Connect, see [Understanding rate limits for APIs and Plans](#).

```
rate-limits:
 Name:
 value: Rate_Limit
 hard-limit: Limit_Toggle
 Name:
 value: Rate_Limit
 hard-limit: Limit_Toggle
burst-limits:
 Name:
 value: Burst_Limit
```

where:

- *Name* is the name of the limit.
- *Rate\_Limit* is your rate limit. It can be for multiples of seconds, minutes, hours, days, or weeks, written as `second`, `minute`, `hour`, `day`, and `week` respectively; do not use the plural forms of the words. Use the syntax: `1/2minute`. If the time unit is singular, then it is not necessary to precede it with a number, for example, `1/minute`. If you do not want to apply a rate limit, set *Rate\_Limit* as `unlimited`.
- *Limit\_Toggle* must be `true` or `false`. If `true`, API calls by a developer will fail if the rate limit is exceeded. This step is not necessary if you have set *Rate\_Limit* to `unlimited`.
- *Burst\_Limit* is your burst limit. It can be for multiples of seconds or minutes, written as `second` or `minute`; do not use the plural forms of the words.

Note:

- Applying a rate limit at the Plan level creates a default rate limit that is shared across all the operations within the Plan. If you need to set specific rate limits for specific operations, you must set these within the operations themselves and these settings will override the setting at the Plan level.
- The test application that is used by the API Manager test tool is not subject to rate limits if you have enabled automatic subscriptions for the Catalog in which you are testing. For more information, see [Working with Catalogs](#)

4. Under your Plan, specify which APIs are to be included.

Reference APIs by name and version, as follows:

```
apis:
 API_1_nameAPI_1_version: {}
 API_2_nameAPI_2_version: {}
```

where

- the *API\_n\_name* variables are the case-sensitive names of your APIs.
- the *API\_n\_version* variables are the versions of your APIs.

5. Optional: If you want to include only a subset of an API's operations, list the operations that are to be included; use the following syntax:

```
apis:
 API_nameAPI_version:
 operations:
 - operation: Operation_1_Verb
 path: Operation_1_Path
 - operation: Operation_2_Verb
 path: Operation_2_Path
```

where

- the *Operation\_n\_Path* variables are the paths of operations that is to be included. The dash is required for each new operation.
- the *Operation\_n\_Verb* variables are the appropriate REST verbs for the operations.

6. Optional: If you want to specify multiple rate limits for a single operation; use the following syntax:

```
apis:
 API_nameAPI_version:
 operations:
 - operation: Operation_Verb
 path: Operation_Path
 rate-limits:
 Name:
 value: Operation_Limit
 hard-limit: Operation_Limit_Toggle
 Name:
 value: Operation_Limit
 hard-limit: Operation_Limit_Toggle
```

where:

- *Operation\_Limit* is the rate limit that you want to apply to your operation. It can be for multiples of seconds, minutes, hours, days, or weeks, written as `second`, `minute`, `hour`, `day`, and `week` respectively; do not use the plural forms of the words. Use the syntax: `1/2minute`. If the time unit is singular, then it is not necessary to precede it with a number, for example, `1/minute`. If you do not want to apply a rate limit, set *Operation\_Limit* as `unlimited`.
- *Operation\_Limit\_Toggle* must be `true` or `false`. If `true`, API calls by a developer will fail if the rate limit is exceeded. This step is not necessary if you have set *Operation\_Limit* to `unlimited`.

## Results

You have described the Plans that are to be included in your Product. Your Plans section should be similar to the following scenarios.

If you have enforced a rate limit for your Plan and not specified individual operations, your Plans section should have the following form:

```
plans:
 Plan_Name:
```

```

title: Plan Title
description: Plan Description
approval: Approval_Toggle
rate-limits:
 Name:
 value: Limit
 hard-limit: Limit_Toggle
apis:
 API_1_nameAPI_1_version: {}
 API_2_nameAPI_2_version: {}

```

If you have enforced rate limits at the Plan level and want to include two operations, with a separate rate limit one for one of the operations, your Plans section should have the following form:

```

plans:
 Plan Name:
 title: Plan Title
 description: Plan Description
 approval: Approval_Toggle
 rate-limits:
 Name:
 value: API_Limit
 hard-limit: API_Limit_Toggle
 burst-limits:
 Name:
 value: Burst_limit
 apis:
 API_nameAPI_version:
 operations:
 - operation: Operation_1_Verb
 path: Operation_1_Path
 rate-limits:
 Name:
 value: Operation_Limit
 hard-limit: Operation_Limit_Toggle
 - operation: Operation_2_Verb
 path: Operation_2_Path

```

In both examples, variables are as in the previous steps and the indentation must be as presented.

The following example shows a complete sample Plans section:

```

plans:
 default:
 title: Default Plan
 description: Default Plan
 approval: false
 rate-limits:
 default:
 value: 100/hour
 hard-limit: false
 my-plan:
 title: my_plan
 rate-limits:
 Default rate-limit:
 value: 50/1hour
 hard-limit: false
 burst-limits:
 Default burst-limit:
 value: 10/1second
 apis:
 api21.0.0:
 operations:
 - operation: get
 path: /path2
 api31.0.0:
 operations:
 - operation: get
 path: /path3
 rate-limits:
 limit3:
 value: 10/1second
 hard-limit: true
 limit4:
 value: 100/1hour
 hard-limit: true
 api11.0.0: {}

```

---

## An example YAML representation of a Product

Products in IBM® API Connect can be represented by using a YAML file in a similar fashion to how APIs can be represented by using OpenAPI.

The following code describes a complete Product, with sample values.

```

info:
 version: 1.0.0
 title: SampleProduct
 name: SampleProduct
 summary: This is a sample product
gateways:
 - datapower-gateway

```

```

product: 1.0.0
visibility:
 view:
 enabled: true
 type: public
 tags: []
 orgs: []
 subscribe:
 enabled: true
 type: authenticated
 tags: []
 orgs: []
plans:
 default:
 title: Default Plan
 description: Default Plan
 approval: false
 rate-limits:
 default:
 value: 100/hour
 hard-limit: false
 newplan:
 title: NewPlan
 rate-limits:
 low:
 value: 10/1hour
 hard-limit: true
 burst-limits:
 Default burst-limit:
 value: 10/1second
apis:
 SampleAPI:
 name: 'Sampleapi:1.0.0'

```

## Using `x-ibm-languages` to create multilingual API and Product documentation

Create multilingual API and Product documentation by using the `x-ibm-languages` extension in your API and Product OpenAPI definitions.

You can scale your API initiatives to a global user base, while still maintaining a single API definition. Translations are custom configured and controlled, so that they can be tailored to both your API documentation and the API consumer. The extensions that are used in API Connect are added directly to the YAML file as `x-ibm-languages`. These extensions are then used in the Developer Portal, so that whatever language the Developer Portal is set to reflects the translations that are contained in the API and Product definitions.

Add the following syntax into your API or Product YAML file at every point that you require some translated text:

```

x-ibm-languages:
 item_name:
 language_code: translated_text

```

Where:

- `item_name` is the name of the item that you want to translate, for example `summary`.
- `language_code` is the ISO code for the translation language. Supported languages with their ISO codes are listed in the following table.

Table 1. List of supported language codes

Language	Code
Chinese, simplified	zh_cn
Chinese, traditional	zh_tw
Dutch	nl
English	en
French	fr
German	de
Italian	it
Japanese	ja
Korean	ko
Portuguese	pt
Spanish	es
Turkish	tr

- `translated_text` is the translated text that you want to be displayed for the item in the Developer Portal.

Note: The Developer Portal automatically assumes that the default language in the OpenAPI definition is English. If you want to use a different default language in the definition, you must provide both the English translation and the default language text in the `x-ibm-languages` extension sections of your OpenAPI definition, by using the following syntax:

```

x-ibm-languages:
 item_name:
 en: English translated text
 default_language_code: default_language_text

```

For an example of how to write an OpenAPI YAML file with French as the default language, see the [Examples](#) section.

## Examples

An example API OpenAPI YAML file that contains the `fr` language extension, where English is the default language:

```
swagger: '2.0'
info:
 x-ibm-name: climbing-weather-api
 title: Climbing Weather API
 version: 1.0.0
 x-ibm-languages:
 title:
 fr: Climat d'escalade
schemes:
 - https
host: $(catalog.host)
consumes:
 - application/json
produces:
 - application/json
x-ibm-configuration:
 assembly:
 execute:
 - invoke:
 target-url: 'https://1234.com'
 title: 3 day forecast invocation
 cache-response: time-to-live
 cache-key: $(request.search)
 gateway: datapower-gateway
 enforced: true
 testable: true
 phase: realized
 cors:
 enabled: true
paths:
 /weather/forecast:
 get:
 summary: Retrieve the 3 day forecast for a location
 description: Retrieve the locations weather forecast descriptions for the next 3 days and nights
 operationId: getWeather
 x-ibm-languages:
 summary:
 fr: Récupérer la prévision de 3 jours pour un emplacement
 description:
 fr: Récupérer les descriptions des prévisions météo pour les 3 prochains jours et les nuits
 tags:
 - Weather
 parameters:
 - name: zip
 type: string
 in: query
 description: A 5 number zip code
 x-ibm-languages:
 description:
 fr: Un code postal à 5 numéros
 - name: country
 type: string
 in: query
 description: A 2 letter country code
 x-ibm-languages:
 description:
 fr: Un code de pays de 2 lettres
 - name: lat
 type: string
 in: query
 description: A latitude value between -90 and 90
 x-ibm-languages:
 description:
 fr: Une valeur de latitude entre -90 et 90
 - name: lon
 type: string
 in: query
 description: A longitude value between -180 and 180
 x-ibm-languages:
 description:
 fr: Une valeur de longitude entre -180 et 180
 responses:
 '200':
 description: Success
 x-ibm-languages:
 description:
 fr: Succès
 '400':
 description: Bad Request
 x-ibm-languages:
 description:
 fr: Mauvaise Demande
 '408':
 description: Request Timeout
 x-ibm-languages:
 description:
 fr: Délai de délai de demande
 '500':
 description: Internal Server Error
 x-ibm-languages:
 description:
 fr: Erreur Interne du Serveur
basePath: /
```



```

tags:
 - name: Weather
 description: Sample API to get weather forecast data
 x-ibm-languages:
 description:
 fr: Exemple d'API pour obtenir des données météorologiques
securityDefinitions:
 client-secret:
 type: apiKey
 description: ''
 in: header
 name: X-IBM-Client-Secret
 client-id:
 type: apiKey
 description: ''
 in: header
 name: X-IBM-Client-Id
security:
 - client-secret: []
 client-id: []

```

An example Product OpenAPI YAML file that contains the `fr` language extension, where English is the default language:

```

product: 1.0.0
info:
 name: climbing-weather
 title: Climbing Weather
 version: 1.0.0
 description: This is a product about weather.
 x-ibm-languages:
 title:
 fr: Météo traduite
 description:
 fr: C'est un produit sur la météo.
 termsOfService:
 fr: Quid ergo aliud intellegetur en francais.
 contact:
 name: Ralph Renli
 email: ralph.renli@example.com
 url: 'https://weather.example.com/climbing/info'
 license:
 name: MIT License
 url: 'https://choosealicense.com/licenses/mit/'
 x-ibm-languages:
 name:
 fr: License de MIT
 termsOfService: Quid ergo aliud intellegetur nisi uti ne quae pars naturae neglegatur? Quis est tam dissimile homini. De
 quibus cupio scire quid sentias.
 categories:
 - Portal/Testing/Language
 - Portal/Testing/Language/UTF8
 - Portal/Testing/Weather
 visibility:
 view:
 enabled: true
 type: public
 tags: []
 orgs: []
 subscribe:
 enabled: true
 type: authenticated
 tags: []
 orgs: []
apis:
 climbing-weather:
 id: 593f972be4b06fb4e0dce879
plans:
 a-call-a-day:
 title: A call a day
 description: "One call every day. That's it!"
 x-ibm-languages:
 title:
 fr: "Un appel par jour"
 description:
 fr: "Un appel tous les jours. C'est tout!"
 apis: {}
 rate-limits:
 One Call Per Day:
 hard-limit: true
 value: 1/1day
 ten-calls-per-day:
 title: 10 Calls a day
 description: '10 times more calls than the next best competitor plan!'
 x-ibm-languages:
 title:
 fr: "10 appels par jour"
 description:
 fr: "10 fois plus d'appels que le prochain meilleur plan concurrent!"
 apis: {}
 rate-limits:
 10 calls per day:
 hard-limit: true
 value: 10/1minute
 11-calls-every-day:
 title: 11 Calls a day
 description: 'Need just one more call? Then this is the plan for you!'

```

```

x-ibm-languages:
 title:
 fr: "11 appels par jour"
 description:
 fr: "Vous n'avez besoin que d'un appel supplémentaire? Alors c'est le plan pour vous!"
 apis: {}
 rate-limits:
 11 calls per day:
 hard-limit: true
 value: 11/day
25-calls-per-day:
 title: 25 Calls per day
 description: So you want even more calls?
 apis: {}
 x-ibm-languages:
 title:
 fr: "25 appels par jour"
 description:
 fr: "Vous voulez donc encore plus d'appels?"
 rate-limits:
 25 calls per day:
 hard-limit: true
 value: 25/day
100-calls-per-day:
 title: 100 Calls every day
 description: 'You get 100 calls each and every day!'
 x-ibm-languages:
 title:
 fr: "100 appels par jour"
 description:
 fr: "Vous recevez 100 appels chaque jour"
 apis: {}
 rate-limits:
 100 Calls per day:
 hard-limit: true
 value: 100/day

```

An example of how to write an OpenAPI YAML file with French as the default language:

```

info:
 name: climat-d-escalade
 title: Climat d'escalade
 version: 1.0.0
 x-ibm-languages:
 title:
 en: Climbing Weather API
 fr: Climat d'escalade
 ...

```

This example shows that both the default French language, and the English language translation, must be provided in the `x-ibm-languages` section.

## Related information

- [Expand your API Initiatives Globally with Multi-Lingual Support of API and Product Definitions](#)

## Using `x-example` to control the examples displayed in the Developer Portal

OpenAPI does not support the use of `example` attributes on parameters, only on request and response objects and their properties. To allow API Developers to be able to control the examples displayed in the portal you can use `x-example` in OpenAPI parameters.

In this example, you cannot use `example` as it isn't supported by the OpenAPI spec, but you can use `x-example`.

```

- name: city
 in: query
 required: false
 type: string
 maxLength: 50
 description: "Location"
 x-example: "New York"

```

An `example` attribute is honored where set and allowed by the spec. If `example` is not found, `default` is honored where set and allowed by the spec. An `x-example` attribute is honored if found, if not found, a random example to match the schema is generated.

If the spec allows you to use `example`, then you should use it in preference to `x-example`.

## Using `x-embedded-doc` to add additional documentation to products and APIs

You can use `x-embedded-doc` to add additional documentation to a product or an API as part of their definition.

You can embed additional documentation pages within the api swagger documentation. These pages can be base64 encoded strings or markdown strings of the html content of the page. These pages show up in the navigation, on the left side, underneath the overview for that product or API, and above the paths. Any styling that is needed to theme the documentation can be included in the Developer Portal custom theme.

## Embedded docs for products and APIs

You can embed documentation for individual products and APIs by specifying an additional `x-embedded-doc` object in the product or API specification.

Field name	Type	Description
name	string	Required, the name of the embedded document
title	string	Required, The display title of the embedded document
format	string	Optional, the format of the embedded document content value. Values are <code>b64html</code> or <code>md</code> . Default value is <code>md</code> .
content	string	The base64 encoded html of page, or, the markdown string.
docs	x-embedded-doc	Additional embedded documentation. If no content is specified for parent, then the first child is the content that is shown.

Hierarchies of documents are supported by nesting of further doc properties. Language support is available by using `x-ibm-languages`.

## Examples

Documentation can be added to an api, by using the following JSON:

```
"x-embedded-doc": [
 {
 "name": "introduction",
 "title": "Introduction",
 "format": "md",
 "content": "# some markdown"
 },
 {
 "name": "concepts",
 "title": "Concepts",
 "format": "md",
 "content": "### more markdown"
 },
 {
 "name": "authentication",
 "title": "Authentication",
 "docs": [
 {
 "name": "auth_clientid_secret",
 "title": "Obtaining a Client ID and Secret",
 "format": "md",
 "content": "### even more markdown"
 },
 {
 "name": "auth_bearertoken",
 "title": "Getting and Using a Bearer Token",
 "format": "md",
 "content": "#### yet more markdown"
 }
]
 }
]
```

Alternatively, by using base 64 encoded HTML:

```
"x-embedded-doc": [
 {
 "name": "introduction",
 "title": "Introduction",
 "format": "b64html",
 "content": "PGFydGljbGUgaWQ9ImIudHJvZHVjdGlvbIiY2xhc3M..."
 },
 {
 "name": "concepts",
 "title": "Concepts",
 "format": "b64html",
 "content": "PGFydGljbGUgaWQ9ImNvbmlCHRzIiBjbGFzZC0icGFnZSI..."
 },
 {
 "name": "authentication",
 "title": "Authentication",
 "docs": [
 {
 "name": "auth_clientid_secret",
 "title": "Obtaining a Client ID and Secret",
 "format": "b64html",
 "content": "PGFydGljbGUgaWQ9ImNsaWVudC1pZC1zZWNYZX..."
 },
 {
 "name": "auth_bearertoken",
 "title": "Getting and Using a Bearer Token",
 "format": "b64html",
 "content": "ICA8YXJ0aWNsZSBpZD0iYmVhcmVvLXRva2VuIiBjbGFz..."
 }
]
 }
]
```

Or mixed formats:

```
"x-embedded-doc": [
 {
 "name": "introduction",
 "title": "Introduction",
 "format": "b64html",
 "content": "PGFydGljbGUgaWQ9ImIudHJvZHVjdGlvbIiY2xhc3M..."
 }
]
```

```

},
{
 "name": "concepts",
 "title": "Concepts",
 "format": "md",
 "content": "# some markdown"
}
]

```

An example of some `x-embedded-doc` code and how it might render:

```

x-embedded-doc:
- name: document
 title: Interesting
 format: b64html
 content: >-
 PGFydGljbGUgaWQ9ImNsaWVudC1pZ...=
securityDefinitions:
 api_key:
 type: apiKey
 in: header
 name: api_key

```

[Products / My Doc APIs](#)



Swagger Petstore Tags 1.0.5 ★★★★★

Overview

Interesting

GET /pet/{petId}

POST /pet/{petId}

DELETE /pet/{petId}

POST /pet/{petId}/uploadImage

POST /pet

PUT /pet

GET /pet/findByStatus

GET /pet/findByTags

GET /store/inventory

## Obtaining a Client ID and Secret

---

Each client app that accesses the API Connect REST API must be registered that determine which operations the app may access.

To register a client app, use the apic registrations:create API Connect

**name**  
short name to identify the client

**title**  
display name of the client

**client\_id**  
a generated client ID value

**client\_secret**  
a generated client secret value

**client\_type**  
one of portal, gateway, toolkit, consumer\_toolkit, ui, consumer\_ui, ibm\_cloud

An example of some `x-embedded-doc` code into a product and how it might render:

```

x-embedded-doc:
- name: introduction
 title: Introduction
 format: md
 content: >-
 ## Overview

 ### Philosophy

 Markdown is intended to be as easy-to-read and easy-to-write as is
 feasible. Compellingly facilitate 2.0 intellectual capital for
 cross-platform networks. Dynamically incubate ethical sources after
 optimal technologies. Phosfluorescently restore global users via excellent
 niche markets. Credibly impact cross-media mindshare through strategic
 best practices. Synergistically impact interdependent web services rather
 than unique interfaces.

 Continually reintermediate enabled sources rather than professional
 architectures. Appropriately embrace viral collaboration and idea-sharing

```

whereas granular solutions. Energistically revolutionize competitive paradigms vis-a-vis highly efficient models. Objectively incentivize enterprise customer service through compelling e-markets. Competently grow multifunctional e-markets with business methods of empowerment.

Dramatically iterate equity invested scenarios and focused data. Compellingly reconceptualize next-generation deliverables rather than out-of-the-box architectures. Competently foster cross-unit web services with interactive innovation. Quickly revolutionize enabled communities and alternative content. Globally administrate resource maximizing deliverables whereas corporate e-business.

Progressively develop maintainable leadership.

```
- name: concepts
 title: Concepts
 format: md
 content: '## some more markdown'
- name: moreinfo
 title: Even more info
 docs:
 - name: moreinfo1
 title: More Info 1
 format: md
 content: '### even more markdown'
 - name: moreinfo2
 title: More Info 2
 format: md
 content: '#### yet more markdown'
```

Products /

## My Product with Docs 1.0.0 ★★★★★

Introduction

Concepts

Even more info

More Info 1

More Info 2

APIs and Plans

# Overview

## Philosophy

Markdown is intended to be as easy-to-read and easy-to-write as is feasible. Compellingly facilitate 2.0 intellectual capital for cross-platform networks. Dynamically incubate ethical sources after optimal technologies. Phosfluorescently restore global users via excellent niche markets. Credibly impact cross-media mindshare through strategic best practices. Synergistically impact interdependent web services rather than unique interfaces. Continually reintermediate enabled sources rather than professional architectures. Appropriately embrace viral collaboration and idea-sharing whereas granular solutions. Energistically revolutionize competitive paradigms vis-a-vis highly efficient models. Objectively incentivize enterprise customer service through compelling e-markets. Competently grow multifunctional e-markets with business methods of empowerment. Dramatically iterate equity invested scenarios and focused data. Compellingly reconceptualize next-generation deliverables rather than out-of-the-box architectures. Competently foster cross-unit web services with interactive innovation. Quickly revolutionize enabled communities and alternative content. Globally administrate resource maximizing deliverables whereas corporate e-business. Progressively develop maintainable leadership.

## Related reference

- [Using x-ibm-languages to create multilingual API and Product documentation](#)

## Using x-pathalias to give consistent URLs for products and APIs

You can use `x-pathalias` to assign a URL to a product or an API.

The information section of a YAML representation of a Product or API can include `x-pathalias`. This option can be useful as it allows the same URL to be used on different revisions of your product. It also allows the use of a vanity URL, which might better represent the branding of your site, or make the URL easier to pronounce.

In this example, you set an API called `Climbing weather API` in product `Weather watch` to have a URL of `portalURL/product/weather/api/climbing`.

Product information:

```
info:
 version: 1.0.0
 title: Weather watch
 name: weather-watch
```

```
x-pathalias: weather
```

API information:

```
info:
title: Climbing Weather API
x-ibm-name: climbing-weather-api
version: 1.0.0
x-pathalias: climbing
```

## Related tasks

---

- [Editing an OpenAPI 2.0 API definition](#)

## Related information

---

- [Creating and configuring Catalogs](#)

# Validating the YAML or JSON definition of an API or Product

You can validate a YAML or JSON definition by using the IBM® API Connect developer toolkit.

## Before you begin

---

To complete the steps that are described in this topic, you must have installed the developer toolkit. For more information, see [Installing the toolkit](#).

## Procedure

---

To perform validation by using the developer toolkit, enter the following command:

```
apic validate filename
```

where *filename* is the file name of the API definition file you want to validate.

- Include `--product-only` to validate only a Product definition and not any APIs that it references.
- Include `--no-extensions` to validate only the default OpenAPI section of the API and none of its extensions.

Note:

- If the OpenAPI file that defines your API uses a `$ref` field to reference a fragment of OpenAPI code that is defined in a separate file, the `$ref` field is replaced with the contents of the target file before the draft API is created with the `apic drafts:validate` command. For more information, see [Using \\$ref to reuse code fragments in your OpenAPI files](#).
- Products that contain an API with a Swagger property using `regex` that include lookahead assertions, such as "(?)" cannot be validated or published. An error message is returned. For example:

```
Product has not been published!
The multipart 'openapi' field contains an OpenAPI definition with validation errors.
definitions.properties.pattern Does not match format 'regex' (context: (root).definitions.properties.pattern, line: 0,
col: 0)
400
```

## Related concepts

---

- [Creating an OpenAPI definition file](#)

# Creating and using API and Product definitions templates

You can use template files when creating OpenAPI 2.0 and OpenAPI 3.0 APIs, and when creating Product definitions. Template files are Handlebars templates containing variables of the form `{{variable-name}}` that are substituted with values when you create the API or Product definition.

## Before you begin

---

To complete the steps that are described in this topic, you must have installed the IBM® API Connect developer toolkit. For more information, see [Installing the toolkit](#).

## Procedure

---

1. Create a Product or API definition template, either from scratch, or by copying one of the examples provided in [API and Product definition template examples](#). The template file must have `.hbs` filename extension, and may contain any of the handlebars variables described in [Template variables for API and Product definitions](#).
2. To create a Product or API definition from a template, enter the following command:

```
apic create:[api | product] --template template_file --title [api_title | product_title] options
```

where *template\_file* is the template `.hbs` file to use, *api\_title* or *product\_title* is the title of the API or Product to create, and *options* is any additional command-line options. The path to the template file can be either an absolute path or relative to the location where the command is executed.

When the Product or API definition is created, the value of each command-line option is substituted for the corresponding handlebars template variable. For example, the value of the required `--title` option is substituted for the `info.title` field in the template file. The command creates an API or Product definition YAML file with the name specified in the `--name` option. If you don't supply the `--name` option, the command derives the name of the API or Product YAML file from the specified title by down-casing the title and replacing spaces with dashes.

## Related reference

---

- [Template variables for API and Product definitions](#)
- [API and Product definition template examples](#)

## Working with global policies

Use global policies to configure policy assemblies that are called just before, or just after, each of your API assemblies is called, or when an error is thrown. You can upload global policies into each of the gateway services in your Catalogs, and then designate, for each gateway service, which global policy should be called before an API assembly is called, after an API assembly is called, or if an error occurs when an API assembly is called. The designated global policies are applied to all the APIs that are deployed to the associated gateway service.

## Prerequisites

---

IBM® DataPower® Gateway version that is compatible with your deployed version of API Connect.

To complete the global policy management tasks described in this topic, you must be the owner of the provider organization, or be assigned a role that has the **Settings:Manage** permission. The pre-supplied Administrator role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see [Creating custom roles](#).

## Overview

---

For each of the gateway services in a Catalog, a global policy that you want to be called before any API assembly is called is designated as a *pre-request* global policy, and a global policy that you want to be called after any API assembly is called is designated as a *post-response* global policy. A policy that you want to be called when an error is thrown is called an *error* global policy. You can upload as many global policies into a gateway service as you want, but you can designate at most one global policy to be pre-request, one to be post-response, and one to be an error, at any one time.

Note:

- If you are using the DataPower Gateway (v5 compatible), there can be only one **proxy** policy across the combined pre-request, API, and post-response assembly flows.
- Catalog properties, as described in [Creating and configuring Catalogs](#), and API properties, as described in [Setting API properties](#), are not supported with global policies. Therefore, if you use such properties in a global policy, they are **not** replaced with the values specified in the API or Catalog property definitions.

A global policy is defined in a .yaml file. The global policy file specifies the policy assembly that you want to call. A global policy file has the following structure:

```
global-policy: 1.0.0

info:
 name: global_policy_name
 title: global_policy_title
 version: global_policy_version

gateways:
 - gateway_type

assembly:
 - policy_assembly_to_be_called
```

where *gateway\_type* has the value `datapower-gateway` if you want to use the global policy in a DataPower Gateway (v5 compatible), and `datapower-api-gateway` if you want to use the global policy in a DataPower API Gateway. For more information on the different types of gateway, see [API Connect gateway types](#).

Note: Error global policies are supported only by the DataPower API Gateway, not by the DataPower Gateway (v5 compatible).

The **assembly** section for a pre-request or post-response global policy can contain any valid API assembly flow. For more information, see the [execute](#) section.

The **assembly** section for an error global policy can only contain a **catch** extension in the assembly, an **execute** section is not valid. For details of the **catch** extension, see [catch](#).

## Sample global policy .yaml file for pre-request or post-reponse

---

```
global-policy: 1.0.0

info:
 name: sample-policy
 title: Sample Policy
 version: 2.0.0

gateways:
 - datapower-api-gateway

assembly:
 execute:
 - invoke:
 target-url: http://myhost.com/mytarget
 version: 2.0.0

 catch:
 - errors:
```

```

- ConnectionError
execute:
- activity-log:
 title: activity-log
 content: activity
 error-content: payload
 version: 2.0.0

```

## Sample error global policy .yaml file

Note: An error global policy .yaml file can contain only a `catch` extension in the assembly.

```

global-policy: 1.0.0

info:
 name: sample-error-policy
 title: Sample Error Policy
 version: 2.0.0

gateways:
- datapower-api-gateway

assembly:
 catch:
 - errors:
 - ConnectionError
 execute:
 - activity-log:
 title: activity-log
 content: activity
 error-content: payload
 version: 2.0.0
 - errors:
 - SOAPError
 execute:
 - xslt:
 version: 2.0.0
 title: xslt
 input: true
 source: |-
 <!-- Placeholder -->
 <xsl:stylesheet xmlns:xsl=''/>

```

## Logging in to the Management server

You work with global policies by using developer toolkit CLI commands. Before you can run the commands, you must log in to your management server from the CLI. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```

apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap

```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details of the `apic login` command, see [Logging in to a management server](#).

## Uploading a global policy to a gateway service

You upload a global policy to a gateway service by using the `apic global-policies:create` command. The command has the following format:

```
apic global-policies:create --catalog catalog_name --configured-gateway-service gateway_service_name --org organization_name --server mgmt_endpoint_url --scope scope [--space space] filename
```

where:

- `catalog_name` is the value of the `name` property of the Catalog that contains the gateway service to which you want to upload the global policy.
- `gateway_service_name` is the value of the `name` property of the gateway service.
- `organization_name` is the value of the `name` property of the provider organization that contains the Catalog.
- `mgmt_endpoint_url` is the platform API endpoint URL.
- `scope` has one of the following values:
  - `catalog` if the Catalog does not have Spaces enabled.
  - `space` if the Catalog has Spaces enabled. If you specify `space` for the `--scope` parameter you must also supply the `--space` parameter to specify the Space that contains the gateway service.



- (optional) *space* is the name of the Space that contains the gateway service. The `--space` parameter is required if the Catalog has Spaces enabled, in which case you must also include `--scope space` in the command.
- *filename* is the name of the global policy .yaml file that you want to upload.

Note: If Spaces are enabled in a Catalog, a global policy that you apply to one Space is applied to **all** Spaces; you cannot apply a global policy to an individual Space in the Catalog. Any subsequent updates are also applied to all Spaces.

You can confirm the global policies that have been uploaded by using the `apic global-policies:list-all` command, which has the following format:

```
apic global-policies:list-all --catalog catalog_name --configured-gateway-service gateway_service_name --org organization_name
--server mgmt_endpoint_url --scope scope [--space space]
```

For reference details of the `apic global-policies` commands, see [apic global-policies](#).

## Designating the pre-request global policy

To designate the pre-request policy for a gateway service, complete the following steps:

1. Write the URL of the required global policy to a .yaml file by using the following command:

```
apic global-policies:get --catalog catalog_name --configured-gateway-service gateway_service_name --org organization_name
--server mgmt_endpoint_url --scope scope [--space space] policy_name:policy_version --fields url
```

where *policy\_name:policy\_version* specifies the name and version number of the required global policy, as defined in the global policy .yaml file that was originally uploaded to the gateway service.

The value of the URL is written to a file called GlobalPolicy.yaml.

2. Edit the GlobalPolicy.yaml file and replace the string `url` with `global_policy_url`. The resulting file has the following format:

```
global_policy_url: >-
 https://server_host_name/api/catalogs/catalog_id/configured-gateway-services/gateway_service_id/global-
 policies/policy_id
```

3. Designate the global policy to be the pre-request policy by using the following command:

```
apic global-policy-prehooks:create --catalog catalog_name --configured-gateway-service gateway_service_name --org
organization_name --server mgmt_endpoint_url --scope scope [--space space] GlobalPolicy.yaml
```

To replace the designated pre-request global policy with another policy, repeat steps [1](#) and [2](#), then use the `apic`

`global-policy-prehooks:update` command. To remove the pre-request global policy designation, use the `apic global-policy-prehooks:delete` command.

For reference details of the `apic global-policy-prehooks` commands, see [apic global-policy-prehooks](#).

By default, whenever an API is called, a series of policies, called the *preflow* policies, are invoked prior to the policy assembly configured in the API definition. However, by using a pre-request global policy you can, if required, fully customize the behavior of these policies, controlling if and when they are invoked, and with what customization. For more information, see [Customizing the preflow policies](#).

## Designating the post-response global policy

To designate the post-response policy for a gateway service, complete the following steps:

1. Write the URL of the required global policy to a .yaml file by using the following command:

```
apic global-policies:get --catalog catalog_name --configured-gateway-service gateway_service_name --org organization_name
--server mgmt_endpoint_url --scope scope [--space space] policy_name:policy_version --fields url
```

where *policy\_name:policy\_version* specifies the name and version number of the required global policy, as defined in the global policy .yaml file that was originally uploaded to the gateway service.

The value of the URL is written to a file called GlobalPolicy.yaml.

2. Edit the GlobalPolicy.yaml file and replace the string `url` with `global_policy_url`. The resulting file has the following format:

```
global_policy_url: >-
 https://server_host_name/api/catalogs/catalog_id/configured-gateway-services/gateway_service_id/global-
 policies/policy_id
```

3. Designate the global policy to be the post-response policy by using the following command:

```
apic global-policy-posthooks:create --catalog catalog_name --configured-gateway-service gateway_service_name --org
organization_name --server mgmt_endpoint_url --scope scope [--space space] GlobalPolicy.yaml
```

To replace the designated post-response global policy with another policy, repeat steps [1](#) and [2](#), then use the `apic`

`global-policy-posthooks:update` command. To remove the post-response global policy designation, use the `apic global-policy-posthooks:delete` command.

For reference details of the `apic global-policy-posthooks` commands, see [apic global-policy-posthooks](#).

## Designating the error global policy

The error global policy is applied to an error caught in the API pre-request, assembly, and post-response stages unless any of those stages has its own error rule defined for the same error, in which case that error rule takes priority.

To designate the error global policy for a gateway service, complete the following steps:

1. Write the URL of the required global policy to a .yaml file by using the following command:

```
apic global-policies:get --catalog catalog_name --configured-gateway-service gateway_service_name --org organization_name
--server mgmt_endpoint_url --scope scope [--space space] policy_name:policy_version --fields url
```

where `policy_name:policy_version` specifies the name and version number of the required global policy, as defined in the global policy `.yaml` file that was originally uploaded to the gateway service.

The value of the URL is written to a file called `GlobalPolicy.yaml`.

2. Edit the `GlobalPolicy.yaml` file and replace the string `url` with `global_policy_url`. The resulting file has the following format:

```
global_policy_url: >-
 https://server_host_name/api/catalogs/catalog_id/configured-gateway-services/gateway_service_id/global-
 policies/policy_id
```

3. Designate the global policy to be the error policy by using the following command:

```
apic global-policy-errors:create --catalog catalog_name --configured-gateway-service gateway_service_name --org
organization_name --server mgmt_endpoint_url --scope scope [--space space] GlobalPolicy.yaml
```

To replace the designated error global policy with another policy, repeat steps 1 and 2, then use the `apic global-policy-errors:update` command. To remove the error global policy designation, use the `apic global-policy-errors:delete` command.

For reference details of the `apic global-policy-errors` commands, see [apic global-policy-errors](#).

---

## Customizing the preflow policies

By default, whenever an API is called, a series of policies, called the *preflow* policies, are invoked prior to the policy assembly configured in the API definition. However, by using a global policy you can, if required, fully customize the behavior of these policies, controlling if and when they are invoked, and with what customization.

---

### Before you begin

- For details on how to configure and deploy a global policy, see [Working with global policies](#).
- For details on how to avoid authentication errors on a WSDL action, see the IBM troubleshooting tech note, [API Connect HTTP GET ?WSDL retrieval returns "401 - Unauthorized Invalid client id or secret"](#).

---

### Default preflow policies

The following preflow policies are invoked by default:

- `cors`: handles cross-origin resource sharing (CORS) requests for the API; invoked if CORS is enabled in the API definition settings.
- `wsdl`: handles WSDL requests.
- `html-page`: for a GraphQL API, if the GraphQL editor option is enabled, returns the GraphQL HTML page.
- `client-identification`: examines the API key credentials that are carried in the API request and matches the API Plan through which the target API is made available to the client.
- `ratelimit`: enforces the rate limit scheme that is configured for the matching API Plan. When the rate limit is reached, the request is rejected.
- `security`: performs the authentication and authorization checks that are required by the target API and operation. If the security requirement is not fulfilled, the request is rejected.

---

## Configuring a global policy to customize the preflow policies

To customize the preflow policies, complete the following steps:

1. add the following property to the `info` section of your global policy `.yaml` file:

```
mode: mode_type
```

Where `mode_type` is one of the following options:

- `after-builtin` - this means that your preflow policies are applied after the default preflow policies.
- `before-builtin` - this means that your preflow policies are applied before the default preflow policies.
- `full-custom` - this means that you must explicitly add the preflow policies to the assembly of your global policy. You can, however, remove, reposition, or replace any of the preflow policies as required.

2. Configure the policies in your global policy assembly as required.

An `after-builtin` global policy example:

```
global-policy: 1.0.0
info:
 name: foo-example
 title: foo-example
 version: 2.0.0
 mode: after-builtin
gateways:
- datapower-api-gateway
assembly:
 execute:
 - set-variable:
 version: 2.0.0
 title: set-variable
 actions:
 - set: message.headers.X-Flexible-Preflow
 value: 'Hello reboot global policy 1 on catalog 1 this policy will be executed after all default builtins'
 type: string
```

A `before-builtin` global policy example:

```
global-policy: 1.0.0
info:
 name: foo-example
```

```

title: foo-example
version: 2.0.0
mode: before-builtin
gateways:
- datapower-api-gateway
assembly:
execute:
- set-variable:
version: 2.0.0
title: set-variable
actions:
- set: message.headers.X-Flexible-Preflow
value: 'Hello reboot global policy 1 on catalog 1 this policy will be executed before all default builtin
policies'
type: string

```

A full-custom global policy example:

```

global-policy: 1.0.0
info:
name: preflow-4-actions
title: preflow-4-actions
version: 2.0.0
mode: full-custom
gateways:
- datapower-api-gateway
assembly:
execute:
- cors:
version: 2.0.0
title: cors-in-preflow
- wsdl:
title: default-wsdl
- html-page:
title: html-page-in-preflow
output: message
version: 2.0.0
- helloworld:
version: 1.0.0
title: helloworld
- client-identification:
version: 2.0.0
title: ci-in-preflow
- ratelimit:
version: 2.0.0
title: assembly-rt-in-preflow
source: plan-default
- security:
version: 2.0.0
title: sec-in-preflow
- set-variable:
version: 2.0.0
title: set-variable
actions:
- set: message.headers.X-Flexible-Preflow
value: 'Hello reboot with 4 actions'
type: string

```

3. Deploy your global policy as a [pre-request global policy](#).

---

## Reference

Reference information for developing your APIs in API Connect, including context variables, and template variables.

Note: For product-wide reference material, including command-line tool and REST APIs documentation, see [Reference information for IBM® API Connect](#).

- [API Connect context variables](#)  
List of IBM API Connect context variables that you can reference when defining default parameter values in an assembly operation, or by using the `getContext()` function when defining a policy.
- [OAuth context variables](#)  
You can customize the OAuth security flow in a native OAuth provider by adding multiple OAuth policies to the assembly. Each OAuth policy takes input from context variables and writes output to context variables. By using these OAuth context variables you can manipulate the original request, and process the output from an OAuth policy.
- [Error cases supported by assembly catches](#)  
Several error cases that can be returned by the assembly are available to the catch function.
- [API and Product definition template examples](#)  
You can use template files when creating API and Product definitions. Template files are Handlebars templates containing variables of the form `{{variable-name}}` that are substituted with values when you create the API or Product definition. This topic provides the default Handlebar templates used by `apic` to create Products and APIs as examples you can use to copy and customize for your own use.
- [Template variables for API and Product definitions](#)  
You can use template files when creating OpenAPI 2.0 and OpenAPI 3.0 APIs, and when creating Product definitions. Template files are Handlebars templates containing variables of the form `{{variable-name}}` that are substituted with values when you create the API or Product definition.

---

## Related information

- [IBM API Connect overview](#)

# API Connect context variables

List of IBM® API Connect context variables that you can reference when defining default parameter values in an assembly operation, or by using the `getContext()` function when defining a policy.

The following tables are presented:

- [General context variables](#).
- [OAuth context variables - DataPower Gateway \(v5 compatible\)](#).
- [Application certificate context variables](#).
- [API activity logging context variables](#).
- [GraphQL context variables](#).

For more information about implementing an assembly component, see [Including elements in your OpenAPI 2.0 API assembly](#), [Including elements in your OpenAPI 2.0 API assembly](#), and for information about how to reference context variables in API Connect see [Variable references in API Connect](#) and [Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway](#).

For more information about creating a user-defined policy, see [Authoring policies](#).

## General context variables

Note:

- For plan variables (such as `plan.name` or `plan.version`), plan information is available only when the requested operation requires identification and the client passes the authentication check.
- If you deploy your API to the DataPower® Gateway (v5 compatible) then, with the exception of client ID and client secret, the passing of form input as a parameter into an API is not supported. This restriction does not apply if you deploy your API to the DataPower API Gateway.

Table 1. API Connect context variables

Name	Description	Permission
<span>API Gateway only</span> <code>api.catalog.id</code>	The ID of the Catalog in which the API is published.	Read/write
<span>API Gateway only</span> <code>api.catalog.name</code>	The name of the Catalog in which the API is published.	Read/write
<span>API Gateway only</span> <code>api.catalog.path</code>	The path segment that represents this Catalog.	Read/write
<code>api.endpoint.address</code>	The address of the API Gateway endpoint.	Read/write
<code>api.endpoint.hostname</code>	The host name of the API Gateway endpoint, as requested by the application.	Read/write
<span>API Gateway only</span> <code>api.id</code>	The identifier of the API.	Read/write
<code>api.name</code>	The name of the API; this corresponds to the <code>x-ibm-name</code> field in the OpenAPI definition for the API.	Read/write
<code>api.operation.id</code>	The ID of the operation.	Read/write
<code>api.operation.path</code>	The path of the operation.	Read/write
<code>api.org.id</code>	The organization ID of the API provider.	Read/write
<code>api.org.name</code>	The organization short name of the API provider.	Read/write
<code>api.properties.propertyname</code>	The name of a custom API property. Property values are Catalog specific. Note: <ul style="list-style-type: none"><li>• You have write permission to a custom property <b>only</b> from the user interface, <b>not</b> from GatewayScript.</li><li>• To access a Catalog specific property value from GatewayScript, you must refer to the property by using the following syntax:  <code>apim-catalog-name</code>  where <code>catalog</code> is the name of the Catalog, and <code>name</code> is the property name. For example:  <code>var mypropertyvalue = \$(apim-mycatalog-mypropertyname)</code></li></ul>	Read/write
<code>api.root</code>	The API base path.	Read/write
<code>api.type</code>	The API type; REST or SOAP.	Read/write
<code>api.version</code>	The version string of the API.	Read/write
<code>client.app.id</code>	The client ID or application key that is received on the request.	Read/write
<span>API Gateway only</span> <code>client.app.lifecycle-state</code>	The lifecycle status of the calling client application. The possible values are as follows: <ul style="list-style-type: none"><li>• <b>DEVELOPMENT</b></li><li>• <b>PRODUCTION</b> (default)</li></ul>	Read/write

Name	Description	Permission
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> <b>client.app.metadata.key</b>	<p>The string value of an application metadata key, where <i>key</i> is the name of the key.</p> <p>You can add metadata keys to an application by using either the <b>apic apps:create</b> or the <b>apic apps:update</b> command; you include the metadata keys in the configuration file parameter that is passed to the command.</p> <p>For example:</p> <pre>apic apps:create myapp.yaml --server myserver.com --org myorg --catalog mycatalog --consumer-org mycorg</pre> <p>where myapp.yaml contains the following:</p> <pre>name: myapp title: My test application metadata:   key1: value1   key2: value2   key3: value3   key4: value4   key5: value5</pre> <p>You can then retrieve the value of a metadata key in an API assembly policy by using a context variable such as the following:</p> <pre>client.app.metadata.key3</pre> <p>Note that adding application metadata might impact gateway trans performance.</p>	Read/write
<b>client.app.name</b>	The name of the application that is identified as having issued the request.	Read/write
<b>client.app.secret</b>	The client secret that is received in the request.	Read/write
<b>client.app.type</b>	The type of the client application that issued the request.	Read/write
<b>client.org.id</b>	The unique identification key of the organization that owns this application.	Read/write
<b>client.org.name</b>	The name of the organization that owns this application.	Read/write
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> <b>client.result</b>	The result of the client security policy, which is <b>SUCCESS</b> or <b>FAILURE</b> .	Read/write
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> <b>client.third_party.type</b>	The type of user registry used for third-party authentication of the extracted client credentials. The possible values are <b>LDAP</b> and <b>auth-url</b> .	Read/write
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> <b>client.third_party.headers</b>	The array of headers added to the request that was sent to that API authentication URL during third-party authentication.	Read/write
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> <b>client.third_party.response.authenticated</b>	The third-party authentication results. The possible values are as follows: <ul style="list-style-type: none"> <li><b>true</b>: the authentication was successful.</li> <li><b>false</b>: the authentication failed.</li> </ul>	Read/write
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> <b>client.third_party.response.user</b>	The user for third-party authentication.	Read/write
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> <b>client.title</b>	The title for the credentials that are received in the request.	Read/write
<div style="border: 1px solid blue; padding: 2px; display: inline-block;">DataPower Gateway (v5 compatible) only</div> <b>env.name</b>	The name of the Catalog in which the API is published.	Read/write
<div style="border: 1px solid blue; padding: 2px; display: inline-block;">DataPower Gateway (v5 compatible) only</div> <b>env.path</b>	The path segment that represents this Catalog.	Read/write
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> <b>error.message</b>	The message text for the error.	Read/write
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> <b>error.name</b>	The name of the error.	Read-only
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> <b>log</b>	The transaction data gathered in an assembly log action.	Read/write
<b>message.attachments</b>	The array of attachments in a multipart message.	Read/write
<b>message.attachments[].body</b>	The attachment payloads in a multipart message.	Read/write
<b>message.attachments[].headers</b>	The attachment headers in a multipart message.	Read/write
<b>message.body</b>	The payload of the request or response message. Note: The <b>message.body</b> context variable is not supported with <b>getContext ()</b> function. Use the <b>getvariable ()</b> function instead.	Read/write
<b>message.headers.name</b>	The value of the current named header of the message or of the current named header of the root part of a multipart message. The <i>name</i> segment is case-insensitive.	Read/write
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> <b>message.original.headers.name</b>	The value of the original named header of the HTTP message. When the assembly contains an invoke action, the value is automatically populated by the original named header of the response message from the invoked URL. The <i>name</i> segment is case-insensitive.	Read-only
<div style="border: 1px solid green; padding: 2px; display: inline-block;">API Gateway only</div> <b>message.package.headers.name</b>	The value of the current named header of the multipart message. The <i>name</i> segment is case-insensitive.	Read/write
<b>message.status.code</b>	The HTTP status code of the response.	Read/write
<b>message.status.reason</b>	The HTTP reason phrase of the response.	Read/write

Name	Description	Permission
API Gateway only <code>message.variables.name</code>	The value of a property in the message. For example, when <code>myvar</code> is the property in the message, you can retrieve the value of the <code>myvar</code> property by reading <code>message.variables.myvar</code> .	Read/write
<code>plan.name</code>	The name of the plan.	Read/write
<code>plan.id</code>	The unique identifier of the plan.	Read/write
<code>plan.rate-limit</code>	The rate limit of the plan, which is the number of API calls per time interval.	Read/write
API Gateway only <code>plan.rate-limit[index].hard-limit</code>	The hard limit setting for the rate limit scheme that is identified by <code>[index]</code> .	Read/write
API Gateway only <code>plan.rate-limit[index].value</code>	The value of the rate limit scheme that is identified by <code>[index]</code> . The syntax is <code>rate/interval</code> .	Read/write
<code>plan.spaceId</code>	The unique identifier of the space that the plan is contained within.	Read/write
<code>plan.spaceName</code>	The name of the space that the plan is contained within.	Read/write
<code>plan.version</code>	The version number of the plan.	Read/write
<code>request.authorization</code>	The parsed HTTP <code>authorization</code> header.	Read-only
<code>request.body</code>	The payload from the incoming request.	Read-only
<code>request.content-type</code>	Normalized content-type value.	Read-only
<code>request.date</code>	A date object that represents approximately when the request was received by the Gateway.	Read-only
<code>request.headers.name</code>	The value of the original named header of the HTTP request, or the value of the current named header of the root part of a multipart request.  The <code>name</code> segment is case-insensitive.  The request header can be modified only in preflow policies. For more information, see <a href="#">Customizing the preflow policies</a> .	Read/write
API Gateway only <code>request.original.headers.name</code>	The value of the original named header of the HTTP request. This variable is created only when the request header is modified. The <code>name</code> segment is case-insensitive.	Read-only
API Gateway only <code>request.package.headers.name</code>	The value of the named header of the multipart request. The <code>name</code> segment is case-insensitive.	Read-only
API Gateway only <code>request.parameters</code>	You can obtain your incoming parameters from path and query parameters.	
API Gateway only <code>request.parameters.name.locations</code>	An array of strings that specify the parameter types associated with the parameter specified in name. The supported keywords for parameter types are as follows.  <code>formData</code> The parameter is in the body as form data <code>header</code> The parameter is in the request header <code>path</code> The parameter is in the path <code>query</code> The parameter is in the query	Read-only
API Gateway only <code>request.parameters.name.values</code>	An array containing values of the parameter types associated with the parameter specified in name. For example, when the first value is <code>path</code> in <code>request.parameters.myparam.locations[]</code> , the first value in <code>request.parameters.myparam.values[]</code> is the array of path values associated with <code>myparam</code> . Request parameters can be modified only in preflow policies. For more information, see <a href="#">Customizing the preflow policies</a> .	Read/write
API Gateway only <code>request.original.parameters.name.values</code>	An array containing the original values of the parameter types associated with the parameter specified in <code>name</code> . This variable is created only when a request parameter value is modified.	Read-only
<code>request.path</code>	The path section of the <code>request.uri</code> that starts with the base path of the API, including the <code>/</code> character that begins the base path.	Read-only
API Gateway only <code>request.protocol</code>	The protocol that is used to receive the request: <code>http</code> or <code>https</code> .	Read-only
<code>request.querystring</code>	The request query string without the leading question mark.	Read-only
<code>request.search</code>	The request query string with the leading question mark.	Read-only
<code>request.uri</code>	The full HTTP request URI from the application.	Read-only
<code>request.verb</code>	The HTTP verb of this request.	Read-only
API Gateway only <code>session.apiGateway</code>	The gateway that receives the request.	Read-only
API Gateway only <code>session.apiGatewayName</code>	The name of the API gateway as defined in the API Manager.	Read-only
API Gateway only <code>session.clientAddress</code>	The address of the client that sent the request.	Read-only
API Gateway only <code>session.domainName</code>	The name of the domain that the gateway belongs to.	Read-only

Name	Description	Permission
<b>API Gateway only</b> session.globalTransactionID	The global transaction ID in the logs.	Read-only
<b>API Gateway only</b> session.localAddress	The address of the gateway on the DataPower® Gateway.	Read-only
<b>API Gateway only</b> session.timeStarted	The time that the gateway started to process the request.	Read-only
<b>API Gateway only</b> session.transactionID	The transaction ID of the gateway request.	Read-only
system.datetime	Returns a string that represents the current date and time in the system time zone of the gateway.	Read-only
system.time	Returns a string that represents the current time in the system time zone of the gateway.	Read-only
system.time.hour	Returns a number 0 - 23 inclusive, representing the hour of the current time in the system time zone of the gateway.	Read-only
system.time.minute	Returns a number 0 - 59 inclusive representing the minute of the current time in the system time zone of the gateway.	Read-only
system.time.seconds	Returns a number 0 - 59 inclusive representing the seconds of the current time in the system time zone of the gateway.	Read-only
system.date	Returns a string that represents the current date in the system time zone of the gateway.	Read-only
system.date.day-of-week	Returns a number 1 - 7 (Monday to Sunday) inclusive representing the day of the week in the system time zone of the gateway.	Read-only
system.date.day-of-month	Returns a number 1 - 31 representing the day of the month in the system time zone of the gateway.	Read-only
system.date.month	Returns a number 1 - 12 representing the month in the system time zone of the gateway.	Read-only
system.date.year	Returns a four-digit number that represents the year in the system time zone of the gateway.	Read-only
system.timezone	Returns a system time zone ISO 8601 identifier for the gateway, which might include a sign, a two-digit hour, and minutes. For example, -04:00.	Read-only

## OAuth context variables - DataPower Gateway (v5 compatible)

The OAuth context variables described in this section apply only to the DataPower Gateway (v5 compatible). For details of the OAuth context variables that apply to the DataPower API Gateway, see [OAuth context variables](#).

Table 2. OAuth context variables (DataPower Gateway (v5 compatible)).

Note: Most OAuth context variables are available only when IBM API Connect is acting as the OAuth resource server. However, the `oauth.introspect` variables are also available when integrating with third party providers.

Name	Description
oauth.access-token	If the request is authenticated with OAuth, this variable contains the access token string.
oauth.miscinfo	This variable contains information explicitly included in the Authentication URL and Metadata URL headers. For more information, see <a href="#">Authenticate URL</a> .
oauth.not-after	If the request is authenticated with OAuth, this variable contains the date when the token expires.
oauth.not-before	If the request is authenticated with OAuth, this variable contains the date when the token was issued.
oauth.resource-owner	If the request is authenticated with OAuth, this variable contains the name of the resource owner.
oauth.scope	If the request is authenticated with OAuth, this variable contains the scope of this access token.
oauth.introspect.active	Always available to introspection. Boolean value.
oauth.introspect.response	Always available to introspection. Shows the complete current response payload. Example payload value: {"active":true, "client_id", "xxx-xxx", "token_type", "bearer", "scope":"neon"}
Other variables might be available from the third party, in the form of: <code>oauth.introspect.&lt;variable&gt;</code>	Decoding the previous example payload, the following variables are made available for further processing.  <pre> oauth.introspect.client_id: xxx-xxx oauth.introspect.token_type: bearer oauth.introspect.scope: neon </pre>

## Application certificate context variables

The following table describes context variables that are available when a certificate is used to verify access to an API, although these will vary depending on the signature mechanism that is being used; for more information, see the [Internet X.509 Public Key Infrastructure Certificate and CRL Profile specification](#).

Table 3. Application certificate context variables

Name	Description
application.certificate.Base64	Base64 format.
application.certificate.fingerprint	Fingerprint
application.certificate.Version	Version
application.certificate.SerialNumber	Serial number
application.certificate.SignatureAlgorithm	Signing algorithm
application.certificate.Issuer	The issuer of the certificate
application.certificate.Subject	Subject
application.certificate.NotBefore	Not valid before this date
application.certificate.NotAfter	Not valid after this date
application.certificate.SubjectPublicKeyAlgorithm	Algorithm for the subject public key
application.certificate.SubjectPublicKeyBitLength	Length for the subject public key

Name	Description
<code>application.certificate.KeyValue.type</code>	<p>Various context variables that depend on the algorithm and key. The following are possible context variables:</p> <ul style="list-style-type: none"> <li><code>application.certificate.KeyValue.RSAKeyValue.Modulus</code></li> <li><code>application.certificate.KeyValue.RSAKeyValue.Exponent</code></li> </ul>

## API activity logging context variables

If activity logging is enabled for an API, a `log` context variable is created; the `log` context variable contains the data relating to an API execution event. On completion of API execution, the `log` context variable is written to an API event record that is stored for subsequent access by API analytics. For details of the fields contained in the `log` context variable, see [API event record fields](#).

The way in which you enable and configure activity logging depends on the type of gateway you are using, as follows:

- If you are using the DataPower API Gateway, you configure activity logging in the API configuration settings; see [Configuring activity logging](#) (OpenAPI 2.0) or [Configuring activity logging](#) (OpenAPI 3.0).
- If you are using the DataPower Gateway (v5 compatible), you configure activity logging by adding an [Activity Log](#) policy to the API assembly.

The activity logging configuration defines the default content of the `log` context variable, but you can modify it in an API assembly; for example:

- Add your own data values by using a [Set Variable](#) policy.
- Remove or redact data values by using a Redaction policy; see [Redaction - DataPower API Gateway](#) or [Redaction - DataPower Gateway \(v5 compatible\)](#).
- API Gateway only Modify or replace the `log` context variable by using a [Log](#) policy.

## GraphQL context variables

The GraphQL query and response variables generated from API processing and assembly actions are stored in the API context in the `message.attributes.graphql` context variables.

Table 4. GraphQL context variables

Name	Description
<code>message.attributes.graphql</code>	The query and response attributes of the GraphQL message.
<code>message.attributes.graphql.query</code>	The attributes of a GraphQL query, including the query string, operation name, and variables.
<code>message.attributes.graphql.query.query</code>	The GraphQL query string.
<code>message.attributes.graphql.query.variables</code>	The map of GraphQL variables.
<code>message.attributes.graphql.query.operationName</code>	The name of the operation to execute.
<code>message.attributes.graphql.query.operationType</code>	The type of the operation to execute.
<code>message.attributes.graphql.request</code>	The attributes of the GraphQL request.
<code>message.attributes.graphql.request.fieldCost</code>	The weighted sum of the cost of all fields accessed in the query. The cost of each field access is configured in the schema. This count includes fields whose field costs are not set to 0.0.
<code>message.attributes.graphql.request.fieldCounts</code>	The maximum number of times that each field can be retrieved by the given query. This number is equal to the number of times that the resolver responsible for producing the value of the field is required to run. This count includes fields whose field costs are not set to 0.0.
<code>message.attributes.graphql.request.typeCost</code>	The weighted sum of the cost of all types retrieved in the query. The cost of each type is configured in the schema. This count includes types whose type costs are not set to 0.0.
<code>message.attributes.graphql.request.typeCounts</code>	The maximum number of times that a value of each type can be retrieved by the given query. This count includes types whose type costs are not set to 0.0.
<code>message.attributes.graphql.response.fieldCost</code>	The weighted sum of the cost of all fields in the response to the query. The cost of each field access is configured in the schema. If this sum cannot be calculated based on the analysis configuration in the request, the field cost is expressed as the maximum integer value that can be assigned.
<code>message.attributes.graphql.response.fieldCounts</code>	The number of times that each field was retrieved in the response to the query. This number is equal to the number of times that the resolver responsible for producing the value of the field is required to run. If this sum cannot be calculated based on the analysis configuration in the request, the field count is expressed as the maximum integer value that can be assigned.
<code>message.attributes.graphql.response.typeCost</code>	The weighted sum of the cost of all types in the response to the query. The cost of each type is configured in the schema. If this sum cannot be calculated based on the analysis configuration in the request, the type cost is expressed as the maximum integer value that can be assigned.
<code>message.attributes.graphql.response.typeCounts</code>	The number of times that a value of each type was contained in the response to the query. If this sum cannot be calculated based on the analysis configuration in the request, the type count is expressed as the maximum integer value that can be assigned.



Name	Description
<code>message.attributes.parse.GraphQLIntrospection</code>	The introspection type of the assembly GraphQL introspect action. When a GraphQL query is parsed, this context variable is populated with one of the following values: <ul style="list-style-type: none"> <li><code>not-introspection</code></li> <li><code>custom-introspection</code></li> <li><code>default-introspection</code></li> </ul>

## Related concepts

- [Variable references in API Connect](#)



## OAuth context variables

You can customize the OAuth security flow in a native OAuth provider by adding multiple OAuth policies to the assembly. Each OAuth policy takes input from context variables and writes output to context variables. By using these OAuth context variables you can manipulate the original request, and process the output from an OAuth policy.

Note: The OAuth context variables described here apply only to the DataPower® API Gateway. For details of the OAuth context variables that apply to the DataPower Gateway (v5 compatible), see [OAuth context variables - DataPower Gateway \(v5 compatible\)](#).

There are two main types of OAuth context variables:

- [Information variables](#), outputs of an OAuth policy.
- [Processing variables](#), processed during the execution of an OAuth policy.

## Information variables

These context variables are outputs of an OAuth policy. They are not used in the processing of the OAuth action.

### `oauth.result`

The result of the latest action; returns **SUCCESS** or **FAILURE**.

### `oauth.settings.variable_name`

Basic settings from the OAuth provider. The following context variables are available:

```

oauth.settings.allowed_scopes
oauth.settings.access_token_ttl
oauth.settings.authorization_code_ttl
oauth.settings.refresh_token_ttl
oauth.settings.refresh_token_limit
oauth.settings.maximum_consent_ttl

```

### `oauth.executed_components[0].variable_name`

The components that have executed in this transaction, with their result. If a failure occurs then the error and error description will be added. The following context variables are available:

```

oauth.executed_components[0].result
oauth.executed_components[0].type
oauth.executed_components[0].error_description
oauth.executed_components[0].error

```

The possible values for the `type` variable are as follows:

```

ValidateRequestComponent
GenerateAZCodeComponent
VerifyAZCodeComponent
VerifyRefreshTokenComponent
GenerateAccessTokenComponent
IntrospectTokenComponent
RevokeTokenComponent
CollectMetaDataComponent

```

The following example shows a corresponding JSON extract from the OAuth context:

```

"executed_components": [
 {
 "type": "ValidateRequestComponent",
 "result": "SUCCESS"
 },
 {
 "type": "GenerateAccessTokenComponent",
 "result": "SUCCESS"
 }
]

```

### `oauth.verified_access_token.variable_name`

The token that was verified in the security requirement for a native OAuth provider. The following context variables are available:

```

oauth.verified_access_token.access_token
oauth.verified_access_token.client_id
oauth.verified_access_token.consented_on
oauth.verified_access_token.consented_on_text

```

```
oauth.verified_access_token.grant_type
oauth.verified_access_token.misc_info
oauth.verified_access_token.not_after
oauth.verified_access_token.not_after_text
oauth.verified_access_token.not_before
oauth.verified_access_token.not_before_text
oauth.verified_access_token.one_time_use
oauth.verified_access_token.resource_owner
oauth.verified_access_token.scope
```

#### `oauth.third_party.variable_name`

The token that was verified in the security requirement for a third party OAuth provider. Provides the following information:

- The response headers from the external, third-party OAuth provider.
- The body of response from the external, third-party OAuth provider.
- Whether the response is cached. When cached, the value is `true`. When not cached, the value is `false`.

The following context variables are available:

```
oauth.third_party.headers
oauth.third_party.response
oauth.third_party.cached
```

#### `oauth.code.variable_name`

The code that was generated during the execution of a component of type `GenerateAZCodeComponent`. The following context variables are available:

```
oauth.code.client_id
oauth.code.code
oauth.code.redirect_uri
oauth.code.resource_owner
oauth.code.scope
```

#### `oauth.token.variable_name`

The token that was generated during the execution of a component of type `GenerateAccessTokenComponent`. The following context variables are available:

```
oauth.token.token_type
oauth.token.access_token
oauth.token.scope
oauth.token.expires_in
oauth.token.consented_on
oauth.token.redirect_uri
oauth.token.resource_owner
oauth.token.client_id
oauth.token.refresh_token
oauth.token.refresh_token_expires_in
oauth.token.refresh_token_count
```

#### `oauth.introspect.variable_name`

The token that was introspected during execution of a component of type `IntrospectTokenComponent`. The following context variables are available:

```
oauth.introspect.active
oauth.introspect.scope
oauth.introspect.client_id
oauth.introspect.resource_owner
oauth.introspect.token_type
oauth.introspect.grant_type
oauth.introspect.ttl
oauth.introspect.expires
oauth.introspect.expires_text
oauth.introspect.iat
oauth.introspect.not_before
oauth.introspect.not_before_text
oauth.introspect.consented_on
oauth.introspect.consented_on_text
oauth.introspect.one_time_use
```

#### `oauth.external_manager.variable_name`

Token management when using an external token manager. Provides the following information:

- The response headers from the external management server.
- The response JSON body from the external management server.
- When caching is enabled, whether the response is found in the cache or set in the cache. When caching is not enabled, that the response is not set in the cache.

The following context variables are available:

```
oauth.external_manager.headers
oauth.external_manager.response
oauth.external_manager.cached
```

## Processing variables

These context variables are processed during the execution of an OAuth policy in your API assembly. If they are set prior to any OAuth policy then the value overrides what was sent in the request. They can also be modified between OAuth policies to manipulate the next component that executes. The following context variables are available:

```
oauth.processing.assertion
oauth.processing.client_id
oauth.processing.client_secret
oauth.processing.grant_type
oauth.processing.redirect_uri
oauth.processing.scope
oauth.processing.response_type
oauth.processing.state
```

```

oauth.processing.resource_owner
oauth.processing.refresh_token
oauth.processing.code
oauth.processing.token
oauth.processing.token_type_hint
oauth.processing.nonce
oauth.processing.max_age
oauth.processing.oidc_values_requested
oauth.processing.id_token_requested
oauth.processing.oidc_signing_algorithm
oauth.processing.code_challenge
oauth.processing.code_challenge_method
oauth.processing.code_verifier

oauth.processing.metadata.access_token
oauth.processing.metadata.payload
oauth.processing.metadata.azcode_miscinfo

oauth.processing.verified_code.client_id
oauth.processing.verified_code.resource_owner
oauth.processing.verified_code.misc_info
oauth.processing.verified_code.scope
oauth.processing.verified_code.is_verified
oauth.processing.verified_code.nonce

oauth.processing.verified_refresh_token.client_id
oauth.processing.verified_refresh_token.resource_owner
oauth.processing.verified_refresh_token.misc_info
oauth.processing.verified_refresh_token.scope
oauth.processing.verified_refresh_token.refresh_token_count
oauth.processing.verified_refresh_token.is_verified
oauth.processing.verified_refresh_token.one_time_use
oauth.processing.verified_refresh_token.grant_type

```

The following example shows the OpenAPI source code for a `gateway-script` policy that executes before an OAuth policy in your API assembly and adds a custom scope to the request:

```

// Add another custom scope to the request
let scope = context.get("request.parameters.scope.values[0]");
if (scope)
 context.set("oauth.processing.scope", scope + " custom");

```

The following example shows the OpenAPI source code for a `gateway-script` policy that between OAuth policies in your assembly and modifies the scope depending on the resource owner:

```

// Check resource owner and modify the scope
let owner = context.get("oauth.processing.resource_owner");
let scope = context.get("oauth.processing.scope");

if (owner === 'admin') {
 context.set("oauth.processing.scope", scope + " admin");
} else {
 context.set("oauth.processing.scope", scope + " customer");
}

```

## Advanced scope context variables

The following context variables are used in OAuth advanced scope checking. They are populated by the values entered in the Endpoint and TLS Client Profile fields for the Application scope check and Owner scope check in the native OAuth provider configuration. Do not change these values as this might result in incorrect advanced scope operation.

For information on scope configuration, see [Configuring scopes for a native OAuth provider](#). For information on advanced scope checking, see [Scope](#).

```

oauth.advscope.app.url
oauth.advscope.app.tls-profile
oauth.advscope.own.url
oauth.advscope.own.tls-profile

```

## Custom error context variables

These context variables are used to stop OAuth processing and specify the respective error details. They can be used before OAuth processing takes place.

```

oauth.custom_error.status.code
oauth.custom_error.status.reason
oauth.custom_error.message
oauth.custom_error.description

```

## Related reference

- [Example - using multiple OAuth policies in an OAuth provider assembly](#)

## Error cases supported by assembly catches

Several error cases that can be returned by the assembly are available to the catch function.

ConnectionError

An error occurred while establishing a connection to another URL.

**JavaScriptError**  
An error occurred while executing JavaScript or GatewayScript in a policy.

**PropertyError**  
An error occurred due to an incorrect property during an invoke call or during the execution of a set-variable policy when an action was not set, add, or clear.

**RedactionError**  
An error occurred during the redaction of a field as part of a redact policy.

**TransformError**  
An error occurred during a transformation policy.

**RuntimeError**  
An otherwise unspecified error occurred.

**BadRequestError**  
An error occurred while trying to access the request. Beginning with Version 10, this is not supported by the default catch; to catch this condition, specify it in your catch definition.

**SOAPError**  
A SOAP Fault response was received as the result of a web service invoke request.

**OperationError**  
An invoke request received a response with an unsuccessful HTTP status code.

**ValidateError**  
An error occurred while trying to schema validate a message payload.

**API Gateway only** **AssemblyRateLimitError**  
The request exceeds the transaction count threshold specified in the API's Plan.

**DataPower Gateway (v5 compatible) only** **UnauthorizedError**  
The API cannot be invoked based on the client ID and/or client secret provided, or id/secret were specified in the wrong location. Beginning with Version 10, this is not supported by the default catch; to catch this condition, specify it in your catch definition.

**DataPower Gateway (v5 compatible) only** **ForbiddenError**  
The application making the API request has been disabled or is not active. Beginning with Version 10, this is not supported by the default catch; to catch this condition, specify it in your catch definition.

## API and Product definition template examples

You can use template files when creating API and Product definitions. Template files are Handlebars templates containing variables of the form `{{variable-name}}` that are substituted with values when you create the API or Product definition. This topic provides the default Handlebar templates used by **apic** to create Products and APIs as examples you can use to copy and customize for your own use.

### Default API definition template (OpenAPI 2.0)

The following example template, which is for an OpenAPI 2.0 API definition, is the default template that developer toolkit uses when you create an API definition. Copy this example template into your own template file (which must have the `.hbs` extension), then edit it for your purposes. Template variables are enclosed by double curly braces, for example `{{name}}`. For information on template variables, see [Template variables for API and Product definitions](#). For more information on Handlebars, see <https://handlebarsjs.com/>.

```
swagger: '2.0'

info:
 x-ibm-name: {{name}}
 title: {{title}}
 version: {{version}}

schemes:
 {{#if schemes}}
 {{#each schemes}}
 - {{this}}
 {{/each}}
 {{else}}
 - https
 {{/if}}
host: {{hostname}}
basePath: {{basepath}}

consumes:
 - application/json
produces:
 - application/json

securityDefinitions:
 clientIdHeader:
 type: apiKey
 in: header
 name: X-IBM-Client-Id
 clientSecretHeader:
 in: "header"
 name: "X-IBM-Client-Secret"
 type: "apiKey"

security:
 -
 clientIdHeader: []
 clientSecretHeader: []

x-ibm-configuration:
```

```

testable: true
enforced: true
gateway: datapower-gateway
cors:
 enabled: true
catalogs:
 apic-dev:
 properties:
 runtime-url: $(TARGET_URL)
 sb:
 properties:
 runtime-url: 'http://localhost:4001'
assembly:
 execute:
 - invoke:
 {{if targeturl}}
 target-url: {{targeturl}}
 {{else}}
 target-url: $(runtime-url)$(request.path)
 {{/if}}

```

paths:

/users:

```

post:
 summary: Create a user
 description: Create a new user
 operationId: userCreate
 externalDocs:
 description: Blah
 url: http://host/docs-about-routes-post
 tags:
 - Users
 responses:
 '201':
 description: 'Success'
 schema:
 $ref: '#/definitions/User'
 default:
 description: 'Unexpected error'
 schema:
 $ref: '#/definitions/Error'

```

```

get:
 summary: User list
 description: Get a list of users
 operationId: userList
 externalDocs:
 description: Blah
 url: http://host/docs-about-routes-post
 tags:
 - Users
 responses:
 '200':
 description: 'Success'
 schema:
 $ref: '#/definitions/UserList'
 default:
 description: 'Unexpected error'
 schema:
 $ref: '#/definitions/Error'

```

/users/{user}:

```

get:
 summary: Retrieve the User
 description: Retrieve the User
 operationId: userGet
 tags:
 - Users
 parameters:
 - name: user
 in: path
 description: User id or name
 required: true
 type: string
 responses:
 '200':
 description: 'Success'
 schema:
 $ref: '#/definitions/User'
 default:
 description: 'Unexpected error'
 schema:
 $ref: '#/definitions/Error'

```

```

patch:
 summary: Update User
 description: Update User
 operationId: userUpdate
 tags:
 - Users
 parameters:
 - name: user
 in: path
 description: User id or name

```

```

 required: true
 type: string
 - name: payload
 in: body
 description: User to update
 required: true
 schema:
 $ref: '#/definitions/UserUpdate'
 responses:
 '200':
 description: 'Success'
 schema:
 $ref: '#/definitions/User'
 default:
 description: 'Unexpected error'
 schema:
 $ref: '#/definitions/Error'

```

```

delete:
 summary: Delete the User
 description: Delete the User
 operationId: userDelete
 tags:
 - Users
 parameters:
 - name: user
 in: path
 description: User id or name
 required: true
 type: string
 responses:
 '204':
 description: 'Successful delete'
 default:
 description: 'Unexpected error'
 schema:
 $ref: '#/definitions/Error'

```

definitions:

```

User:
 type: object
 additionalProperties: false

```

```

UserUpdate:
 type: object
 additionalProperties: false

```

```

UserList:
 type: object
 additionalProperties: false

```

```

Error:
 type: object
 additionalProperties: false
 properties:
 status:
 type: integer
 message:
 type:
 - string
 - array

```

```

tags:
 - name: Users
 description: Tags on all the user operations
 externalDocs:
 description: External information about Users
 url: http://host/url-of-my-entire-set-of-tag-docs-for-this-tag
 - name: Routes
 description: Tags on all the route operations
 externalDocs:
 description: External information about Routes
 url: http://host/url-of-my-entire-set-of-tag-docs-for-this-tag

```

## OpenAPI 3.0 API definition template

The following example template is for an OpenAPI 3.0 API definition. Copy this example template into your own template file (which must have the `.hbs` extension), then edit it for your purposes. Template variables are enclosed by double curly braces, for example `{{name}}`. For information on template variables, see [Template variables for API and Product definitions](#). For more information on Handlebars, see <https://handlebarsjs.com/>.

```

openapi: '3.0.3'

info:
 x-ibm-name: {{name}}
 version: {{version}}
 title: {{title}}

x-ibm-configuration:
 assembly:
 execute:
 - invoke:
 title: invoke
 version: 2.0.0

```

```

 target-url: 'http://apicsrv01.dp.rtp.raleigh.ibm.com/sample.json'
cors:
 enabled: true
 enforced: true
 phase: realized
 testable: true
 gateway: datapower-api-gateway

servers:
- url: "/oauth-supported-flows"

paths:
 "/getData":
 get:
 summary: Operation Get Branches
 operationId: basicImplicit
 responses:
 default:
 description: Default response

security:
- OAuth:
 - scopePub

components:
 securitySchemes:
 clientIdHeader:
 type: apiKey
 in: header
 name: clientHeader
 clientSecretHeader:
 type: apiKey
 in: header
 name: X-IBM-Client-Secret
 client_id:
 type: apiKey
 in: query
 name: client_id
 client_secret:
 type: apiKey
 in: query
 name: client_secret

```

## OAuth 2.0 API definition template

The following example template is the default template that developer toolkit uses when you create an OAuth 2.0 API definition with the command `apic create:api -template oauth`. Copy this example template into your own template file (which must have the `.hbs` extension), then edit it for your purposes. Template variables are enclosed by double curly braces, for example `{{name}}`. For information on template variables, see [Template variables for API and Product definitions](https://handlebarsjs.com/). For more information on Handlebars, see <https://handlebarsjs.com/>.

```

swagger: "2.0"

info:
 x-ibm-name: {{name}}
 title: {{title}}
 version: {{version}}

schemes:
 {{#if schemes}}
 {{#each schemes}}
 - {{this}}
 {{/each}}
 {{else}}
 - https
 {{/if}}
host: {{hostname}}
basePath: {{basepath}}

securityDefinitions:
 clientID:
 description: "application's client_id"
 in: "query"
 name: "client_id"
 type: "apiKey"

security:
- clientID: []

paths:
 /oauth2/authorize:
 get:
 produces:
 - text/html
 summary: endpoint for Authorization Code and Implicit grants
 description: description
 parameters:
 - name: response_type
 in: query
 description: request an authorization code or or access token (implicit)
 required: true
 type: string

```

```

enum:
 - code
 - token
- name: client_id
 in: query
 description: Application client ID
 required: true
 type: string
- name: scope
 in: query
 description: Scope being requested
 type: string
 required: true
- name: redirect_uri
 in: query
 type: string
 description: URI where user is redirected to after authorization
 required: false
- name: state
 in: query
 type: string
 description: This string will be echoed back to application when user is redirected
 required: false
responses:
 302:
 description: |
 Redirect to the clients redirect_uri containing one of the following
 - **authorization code** for Authorization code grant
 - **access token** for Implicity grant
 - **error** in case of errors, such as the user has denied the request
 200:
 description: An HTML form for authentication or authorization of this request.
security:
- clientID: []

```

```

post:
 consumes:
 - application/x-www-form-urlencoded
 produces:
 - text/html
 summary: submit approval to authorization code or access token
 description: |
 Submit resource owners approval (or rejection) for the OAuth2 Server to issue an
 authorization code or access token to the application.
 security:
 - clientID: []
 parameters:
 - name: client_id
 in: formData
 description: application requesting the access code or token
 required: true
 type: string
 - name: scope
 in: formData
 description: requested scope of this authorization
 required: true
 type: string
 - name: resource-owner
 in: formData
 description: resource owners user name
 required: true
 type: string
 - name: redirect_uri
 in: formData
 description: URI the application is requesting this code or token to be redirected to
 required: true
 type: string
 - name: original-url
 in: formData
 description: URL of the original authorization request
 required: true
 type: string
 - name: dp-state
 in: formData
 description: state information provided in the authorization form
 required: true
 type: string
 - name: dp-data
 in: formData
 description: state information provided in the authorization form
 required: true
 type: string
 #- name: response_type
 # in: formData
 # description:
 # required: true
 # type: string
 responses:
 200:
 description: Cool

```

/oauth2/token:

```

post:
 consumes:
 - application/x-www-form-urlencoded

```



```

produces:
- application/json
summary: Request Access Tokens
description: |
This endpoint allows requesting an access token following one of the flows below:
- Authorization Code (exchange code for access token)
- Client Credentials (2-legged, there isnt resource owner information)
- Resource Owner Password Credentials (2-legged, client provides resource owner name and password)
- Refresh Token (exchange refresh token for a new access code)

```

The table below indicates the required parameters for each specific grant\_type options.  
Empty cells indicate a parameter is ignored for that specific grant type.

Client authentication:

- Confidential clients should authenticate using HTTP Basic Authentication. Alternatively, they may post their client\_id and client\_secret information as a formData parameter.
- Public clients should send their client\_id as formData parameter.

grant_type	code	client_credentials	password	refresh_token
client_id	required*	required*	required*	required*
client_secret	required*	required*	required*	required*
code	required			
redirect_uri	required			
username			required	
password			required	
scope		optional	optional	
refresh_token				required

The implicit grant requests, see /oauth2/authorize.

```
security: []
```

```
parameters:
```

```

- name: grant_type
 in: formData
 description: Type of grant
 type: string
 required: true
 enum:
 - authorization_code
 - password
 - client_credentials
 - refresh_token
- name: client_id
 in: formData
 description: Application client ID, can be provided in formData or using HTTP Basic Authentication
 required: false
 type: string
- name: client_secret
 in: formData
 description: Application secret, must be provided in formData or using HTTP Basic Authentication
 required: false
 type: string
- name: code
 in: formData
 description: Authorization code provided by the /oauth2/authorize endpoint
 required: false
 type: string
- name: redirect_uri
 in: formData
 description: required only if the redirect_uri parameter was included in the authorization request /oauth2/authorize;
their values MUST be identical.
 required: false
 type: string
- name: username
 in: formData
 type: string
 description: Resource owner username
 required: false
- name: password
 in: formData
 type: string
 description: Resource owner password
 required: false
- name: scope
 in: formData
 type: string
 description: Scope being requested
 required: false
- name: refresh_token
 in: formData
 type: string
 description: The refresh token that the client wants to exchange for a new access token (refresh_token grant_type)
 required: false

```

```
responses:
```

```

200:
 description: json document containing token, etc.
 schema:
 $ref: "#/definitions/access_token_response"
400:
 description: json document that may contain additional details about the failure

```

```
x-ibm-configuration:
```

```

testable: true
enforced: true

```

```

phase: "realized"
oauth2:
 client-type: public #or confidential
 scopes:
 scope1: Description 1
 scope2: Description 2
 scope3: Description 3
 grants:
 - application
 - password
 - accessCode
 - implicit

identity-extraction:
 type: default-form #If identity extraction is not there use this form.
 #type: basic
 #type: custom-form #Customer provided form (needs location)
 #type: redirect #Redirects user to authenticate somewhere else
 #custom-form:
 # url: https://example.com/authentication/form
 # tls-profile: tls-profile-1
 #redirect-url: https://example.com/external/form

authentication:
 x-ibm-authentication-url:
 url: https://example.com/auth/url
 tls-profile: tls-profile-4
 #x-ibm-authentication-registry: ldap-1

authorization:
 type: authenticated #If the authorization section is missing this is the default
 #type: default-form
 #type: custom-form
 #custom-form:
 # url: https://example.com/authorization/form
 # tls-profile: tls-profile-2

refresh-token:
 count: 2048 # If this section is missing default is 0
revocation:
 url: ""
 tls-profile: ""

definitions:

access_token_response:
 type: object
 additionalProperties: false
 required:
 - token_type
 - access_token
 - expires_in
 properties:
 token_type:
 enum:
 - bearer
 access_token:
 type: string
 expires_in:
 type: integer
 scope:
 type: string
 refresh_token:
 type: string

```

## Product definition template

The following example template is the default template that developer toolkit uses when you create a Product definition. Copy this example template into your own template file (which must have the `.hbs` extension), then edit it for your purposes. Template variables are enclosed by double curly braces, for example `{{name}}`. For information on template variables, see [Template variables for API and Product definitions](#). For more information on Handlebars, see <https://handlebarsjs.com/>.

```

product: '1.0.0'

info:
 name: {{name}}
 title: {{title}}
 version: {{version}}

{{#isEmpty apis}}
{{else}}
apis:
 {{/isEmpty}}
 {{#each apis}}
 '{{@key}}':
 $ref: {{this}}
 {{/each}}

visibility:
 view:
 type: public
 subscribe:
 type: authenticated

plans:

```

```

default:
 title: Default Plan
 description: Default Plan
 approval: false
 rate-limit:
 value: 100/hour
 hard-limit: false

```

## Related tasks

- [Creating and using API and Product definitions templates](#)

## Related reference

- [Template variables for API and Product definitions](#)

# Template variables for API and Product definitions

You can use template files when creating OpenAPI 2.0 and OpenAPI 3.0 APIs, and when creating Product definitions. Template files are Handlebars templates containing variables of the form `{{variable-name}}` that are substituted with values when you create the API or Product definition.

## Product definition variables

The following table describes the Handlebars template variables that can be used in a product definition file. For more information on Handlebars, see <https://handlebarsjs.com/>. Product definition template files must have a `.hbs` filename extension.

Table 1. Product definition Handlebars template variables

Variable	Type	Description
{{apis}}	Array of string	The APIs to which the product definition refers. After substitution, the array values become the values of <code>apis.routes[n].\$ref</code> fields, for example: <pre> apis:   'routes':     \$ref: apidef.yaml     ... </pre>
{{name}}	String	Value of <code>info.x-ibm-name</code> field.
{{title}}	String	Value of <code>info.title</code> field.
{{version}}	String	Value of <code>info.version</code> field.

## API definition variables

The following table describes the Handlebars template variables that can be used in an API definition file. For more information on Handlebars, see <https://handlebarsjs.com/>. API definition template files must have a `.hbs` filename extension.

Table 2. API definition Handlebars template variables

Variable	Type	Description
{{basepath}}	String	Base path on which the API is served, which is relative to the <code>host</code> .
{{definitions}}	<a href="#">OpenAPI definitions object</a> converted to a YAML string ("stringified").	For a LoopBack API, contains data types that can be consumed and produced by operations. These data types can be primitives, arrays or models.
{{definitionsObj}}	<a href="#">OpenAPI definitions object</a> converted to a YAML string ("stringified").	For a LoopBack API, contains data types that can be consumed and produced by operations. These data types can be primitives, arrays or models.
{{hostname}}	String	Value of the <code>host</code> field.
{{name}}	String	Value of <code>info.x-ibm-name</code> field.
{{paths}}	<a href="#">OpenAPI paths object</a>	For a LoopBack API, contains the relative paths to the individual endpoints. The path is appended to <code>{basePath}</code> to construct the full URL.
{{pathsObj}}	<a href="#">OpenAPI paths object</a>	For a LoopBack API, contains the relative paths to the individual endpoints. The path is appended to <code>{basePath}</code> to construct the full URL.
{{schemes}}	Array of string	Transfer protocol of the API. Values must be one of: "http", "https", "ws", or "wss".
{{targeturl}}	String	Value of <code>x-ibm-configuration.assembly.execute[invoke]</code> . Default is <code>\$(runtime-url)\$(request.path)</code> .
{{title}}	String	Value of <code>info.title</code> field.
{{version}}	String	Value of <code>info.version</code> field.

## Related tasks

- [Creating and using API and Product definitions templates](#)

## Related reference

- [API and Product definition template examples](#)

---

# Managing your APIs

You manage your APIs by using the API Manager user interface of IBM® API Connect. You can also analyze your API usage by using the analytics that are provided and socialize your APIs in a developer portal.

---

## Why use APIs?

Whether you are a business user, an IT user, or an application developer, APIs are increasingly important to your business. You can use an API to publicize your company. The assets, data, or services of your company can be provided to external application developers to expand your enterprise and open new markets.

The API Manager UI provides a solution for companies to manage APIs for private internal APIs, and public external APIs. This on-premises offering provides the capabilities that are required so that you can externalize and manage your services as REST or SOAP APIs.

---

## What do you need to know?

Depending on your role, you can complete different tasks relating to managing your API Catalogs and these are outlined in the documentation. Each task is covered in the order that they are executed.

Each task introduces new features of the API Manager UI as they become relevant to the Catalog that you are building.

For security reasons, your session times out after a period of inactivity.

Before you start, check that the browser you are using is supported and meets the required minimum levels; for details, open the [Detailed system requirements for a specific product](#) page, search for the IBM API Connect product, then select the required version.

The ways in which you can manage your APIs are described in the following subtopics:

- [API management checklist](#)  
A summary, with links, of the key tasks for managing APIs in IBM API Connect.
- [Activating your API Manager user account](#)  
Before you can access the API Manager user interface of IBM API Connect, you must activate the user account that your Cloud Manager administrator invited you to join.
- [Accessing the API Manager user interface](#)  
The IBM API Connect API Manager user interface provides a range of options for working with your APIs and Products, and managing security.
- [Searching for items in API Manager](#)  
Use the search feature in IBM API Connect API Manager to easily locate items such as APIs, Catalogs, Spaces, applications, and subscriptions.
- [Working with Catalogs](#)  
Products must be staged to a Catalog and then published; application developers are able to access the APIs in a Product by being members of consumer organizations to which the Product is made available. In API Connect, you can create multiple Catalogs. Catalogs are useful for separating Products and APIs for testing before you make them available to consumer organizations. The syndication feature in API Connect means that you can also publish a Product to a Space in a Catalog.
- [Using syndication in API Connect](#)  
With the API Connect syndication feature, you can partition your Catalogs into *Spaces*. Each Space is used by a different API provider development team and has its own set of management capabilities relating specifically to the APIs that the associated team publishes to that Space, enabling each team to manage their APIs independently.
- [Working with Products](#)  
In IBM API Connect, Plans and APIs are grouped together in Products.
- [Working with APIs](#)  
For a specific Product, you can work with the APIs that are contained in the Plans in that Product.
- [Working with Plans](#)  
After a Product has been published to a Catalog, you can work with the Plans in that Product.
- [Working with consumer organizations](#)  
Manage the consumer organizations that access your APIs and Plans when their users sign up to use the IBM API Connect Developer Portal.
- [Working with developer applications](#)  
You can work with the applications that have been registered in the Developer Portal associated with a Catalog.
- [Working with application subscriptions](#)  
You can work with the subscriptions to the Plans in all the Products in a specific Catalog. Plan subscriptions are created by application developers in the Developer Portal.
- [Security and authentication](#)  
In API Manager, you can use TLS profiles to secure the transmission of data between the management server and other API Connect subsystems and external services, and also configure user registries to securely authenticate your Catalogs and APIs.
- [API Analytics](#)  
You can use API Connect to filter, sort, and aggregate your API event data. You can present the results within correlated charts, tables, and maps to help you manage service levels, set quotas, establish controls, set up security policies, manage communities, and analyze trends.
- [Administering user access](#)  
If you have permission to administer users in IBM API Connect, you can add and delete users. After users are removed from an organization through deletion, the user account remains in API Manager.
- [Changing your API Manager password](#)  
You can change your API Manager password in IBM API Connect.
- [Resolving login problems by increasing HTTP header size](#)  
You can resolve login problems for the API Manager UI by increasing the maximum HTTP client header size.
- [Reviewing a gateway's processing status](#)  
Review the processing status for the Gateway services that are enabled in your API Connect Catalogs and Spaces.
- [Reference](#)  
Reference information for the API Manager component in IBM API Connect.

---

## Related information

- [IBM API Connect overview](#)
- [Using the Developer Portal](#)

## API management checklist

A summary, with links, of the key tasks for managing APIs in IBM® API Connect.

Task	Description
<a href="#">Activate your API Manager user account</a>	To connect to a Management server using the API Manager user interface, you must be an owner or a member of a provider organization.
<a href="#">Log in to the API Manager user interface</a>	After you have activated your account, you can log in to your Management server from the API Manager user interface.
<a href="#">Configure your Catalogs</a>	APIs are published to a Catalog to make them available to application developers. The URLs for API calls, and for the Developer Portal, are specific to a particular Catalog. By default, a <i>sandbox</i> Catalog is provided for API testing, but you can create your own Catalogs; for example, a production Catalog for hosting APIs that are ready for full use.
<a href="#">Manage the membership of your Catalogs</a>	You can add users to your Catalogs, and assign roles and permissions to control their level of access.
<a href="#">Configure API security</a>	You can configure user registries of various types for authenticating API access, TLS profiles for securing data transmission over HTTPS, and OAuth providers for securing third-party website or application access to APIs.
<a href="#">Manage your Products</a>	Move your Products through their lifecycle, from initially staging a Product to a Catalog, through to publishing to make the Product version available to your application developers, and to eventual retiring and archiving. You can also display analytics information, and control who can see or subscribe to your Products.
<a href="#">Manage user access to your provider organization</a>	You can add users to your provider organization, remove users, and assign roles and permissions.
<a href="#">Partition your Catalogs into Spaces for syndication</a>	With the syndication feature, you can partition your Catalogs into <i>Spaces</i> . Each Space is used by a different API provider development team and has its own set of management capabilities relating specifically to the APIs that the associated team publishes to that Space, enabling each team to manage their APIs independently.

## Activating your API Manager user account

Before you can access the API Manager user interface of IBM® API Connect, you must activate the user account that your Cloud Manager administrator invited you to join.

### Before you begin

The Cloud Manager administrator invited you to join API Connect .

### Procedure

1. Complete the following steps to create your API Manager account:

If the Identity Provider uses LDAP

An invitation email with an activation link is sent. Click the activation link, or paste it in a browser, to log in directly with your user credentials. Upon authentication, the API Connect user record is updated from the backend identity provider.

If the Identity Provider uses a local registry

An invitation email with an activation link is sent. Click the activation link or paste it in a browser. The activation link takes you to a sign up page where you enter your first name, last name and password. Passwords must have a minimum of 8 characters and contain characters from at least three of the four following categories:

- Uppercase letters
- Lowercase letters
- Numbers
- Special characters (for example: ! # \$ %)

Note:

- The email address that you enter on the sign up page must match the email address to which the invitation email was sent, otherwise the account activation fails.
- If you previously had an account that the administrator removed, and they are inviting you again, you must re-activate your account by using the Sign In option on the page, **not** by completing the registration form and using the Sign Up option; attempting to re-register will fail.

If the Identity Provider uses an authentication URL

An invitation email with an activation link is sent. Click the activation link or paste it in a browser. The activation link takes you to a sign up page where you enter your credentials, which then become your user name and password. Upon authentication, the API Connect user record is updated from the backend identity provider.

If multiple user registries are available for selection on the sign up page, then make sure that the correct registry for your API Manager account is selected. You might need to ask your administrator which user registry is appropriate for your account.

For information on configuring user registries and making them available for API Manager login, see [User registries overview](#) and [Selecting user registries for Cloud Manager and API Manager](#).

2. Click Sign up to complete your registration, then click Sign in to open the API Manager login page.

## Results

---

You have created your API Connect account.

## What to do next

---

Log in to the API Manager user interface and, depending on your role, start to work with Catalogs, consumer organizations, APIs and Products. To access the API Manager login page in the future, use the following URL:

`https://host/manager`

where *host* is the fully qualified host name or IP address of the Management server.

## Related concepts

---

- [Working with Products](#)

## Related tasks

---

- [Creating and configuring Catalogs](#)
- [Working with consumer organizations](#)

## Related reference

---

- [Accessing the API Manager user interface](#)

## Related information

---

- [Developing your APIs and applications](#)

---

## Accessing the API Manager user interface

The IBM® API Connect API Manager user interface provides a range of options for working with your APIs and Products, and managing security.

To log in to the API Manager user interface, complete the following steps:

1. In a web browser, enter the following URL:

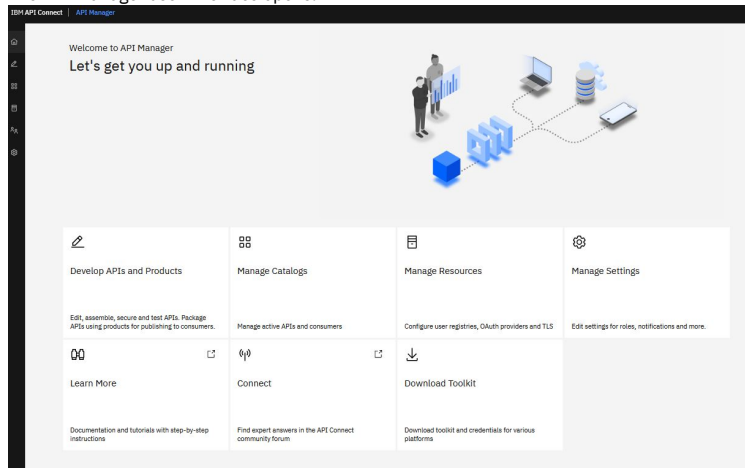
`https://host/manager`

where *host* is the fully qualified host name or IP address of the Management server.

2. If your cloud contains multiple user registries, then select the user registry that contains the login credentials for your API Manager account. You might need to ask your administrator which user registry is appropriate for your account. For information on configuring user registries and making them available for API Manager login, see [User registries overview](#) and [Selecting user registries for Cloud Manager and API Manager](#).

3. Enter your user name and password, then click Sign in.

The API Manager user interface opens:




Depending on your role, you can now start to work with Catalogs, consumer organizations, APIs and Products.

---


## Searching for items in API Manager

Use the search feature in IBM® API Connect API Manager to easily locate items such as APIs, Catalogs, Spaces, applications, and subscriptions.

Search is available from any page in API Manager and can be accessed by typing in the search field in the page banner, or by clicking  Search in the navigation list.

## Using Search

---

The Search feature is available on all pages of the API Manager. To start searching, just type a string into the Search field and press Enter. Whenever you press Enter in the Search field, the Search page displays with results and additional options. If you press Enter in an empty Search field, or click  Search in the navigation list, then the Search page displays a list all items. If you type a string before pressing Enter, the page displays the results from the corresponding search.

Search results include all items that contain the search string in a text field. For example, suppose you searched for "Crafted". The results might look like the following image, with items that don't match the search string exactly, but do include the string in a text field such as a title or a name.

By default, the search returns all types of items that contain the string. Sort results on a particular column by clicking the column header.

When you find an item you want to work with, click the . . . actions menu next to the item's "Modified" date to see what you can do. The list of possible actions depends on the item's type.

You can limit the search results by selecting an item type from the list that displays before the results. Click More to view additional types. If you select a type from the More menu, it replaces a type in the list.

If you want to refine the search further, click Options. On the Options panel, click Catalogs to Search and select which catalogs to search (the default is all catalogs). You can select up to 5 catalogs for a limited search.

You can also choose whether only exact matches (the complete string) are included in results, or items containing similar strings are also included.

---

## Working with Catalogs

Products must be staged to a Catalog and then published; application developers are able to access the APIs in a Product by being members of consumer organizations to which the Product is made available. In API Connect, you can create multiple Catalogs. Catalogs are useful for separating Products and APIs for testing before you make them available to consumer organizations. The syndication feature in API Connect means that you can also publish a Product to a Space in a Catalog.

A Catalog is a staging target, and behaves as a logical partition of the gateway and the Developer Portal. The URL for API calls and the Developer Portal are specific to a particular Catalog. In a typical configuration, an API provider organization uses a development Catalog for testing APIs under development and a production Catalog for hosting APIs that are ready for full use. A common approach is to have a development cloud with a development Catalog, a few test Catalogs and a production cloud that might have its own test Catalog.

You can use a *Space* to partition a Catalog so multiple teams can manage Products and APIs independently in a single Catalog. A Space is conceptually like a sub-catalog, except that Products and APIs in all Spaces in a given Catalog are published to the same developer portal. For more information about Spaces, see [Using syndication in API Connect](#).

Note:

Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: **A user already exists with this email address.**

For more information on using the Developer Portal, see [Developer Portal: Socialize your APIs](#).

- [Creating and configuring Catalogs](#)  
These instructions describe how to create and configure Catalogs in IBM API Connect.
- [Configuring Product visibility in a Catalog or a Space](#)  
How to configure the visibility of Products at the Catalog or Space level.
- [Configuring sub paths for Developer Portal sites](#)  
You can increase the specificity of a IBM API Connect Developer Portal site URL by using sub paths.
- [Managing Catalog membership](#)  
Manage Catalog membership by adding new users to the Catalog and assigning roles to the users.
- [Importing a user-defined policy into a Catalog](#)  
You can make your user-defined policy available to API developers by importing it into an IBM API Connect Catalog, or a Space in a Catalog.
- [Importing an OpenAPI extension into a Catalog](#)  
You can extend the OpenAPI specification by adding either a JSON or YAML extension schema to an API. An extension is imported into a Catalog, or a Space in a Catalog, then added to the API schema.
- [Retaining Version 5 vanity endpoint behavior in a Catalog](#)  
The behavior of vanity endpoints in a Catalog has changed in API Connect Version 10 compared with Version 5. However, you can modify your Catalog settings to retain the Version 5 behavior.

---

## Creating and configuring Catalogs

These instructions describe how to create and configure Catalogs in IBM® API Connect.

### Before you begin

---

You must have Catalog create permissions to complete this task. For more information about permissions, see [API Connect user roles](#).


## Notes:

- As the user who creates the Catalog, you are automatically the Catalog owner and have all Catalog permissions.
- If you configure a new user registry for authenticating users to the Developer Portal, you must also complete the onboarding section in the Catalog, to make the registry available to Developer Portal users. See Step [15](#) for details.
- Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: **A user already exists with this email address.**

## Procedure

---

To create and configure your Catalog, complete the following steps:

1. Log in to API Manager and click  Manage.
2. Create the Catalog.  
You can create the Catalog in either of the following ways:  
Specify the owner, then configure the Catalog:
  - a. Click [Add > Create catalog](#).
  - b. In the Catalog Owner field, select the owner of the Catalog; by default, you are the owner. You can specify any user who is a member of the provider organization.
  - c. Continue from step [3](#) to configure the Catalog.Send an invitation email, with an activation link, to the intended Catalog owner:
  - a. Click [Add > Invite catalog owner](#).
  - b. Enter the email address of the Catalog owner, then click Invite.
  - c. On receipt of the email, the Catalog owner must activate the Catalog and configure it:
    - i. Open the activation link, complete the sign up form, and sign in to the API Manager UI.
    - ii. Click Manage Catalogs, click the Catalog name that was provided on the sign up form, then click the Catalog settings tab.
    - iii. Continue from step [6](#) to configure the Catalog.
3. Enter the Catalog Title. A Name is entered automatically and cannot be changed.  
Note: The value in the Name field is a single string that is used to identify the Catalog in developer toolkit CLI commands. The Title is used for display; you can change the title but the name does not change.  
To view the CLI commands to manage Catalogs, see [apic catalogs](#).
4. Click Create.  
Your new Catalog is displayed on the Manage page.
5. To continue to configure your Catalog, select the Catalog on the Manage page, then click the Catalog settings tab.
6. To configure the general settings for the Catalog, click Overview, then proceed with the following steps:
  - a. By default, the new Catalog is a development Catalog. To use the Catalog in production, set the Production Mode slider control to the On position, then click Confirm.  
In a development Catalog, all publish operations are forced, and any conflicts are resolved automatically. If you republish a previously published Product version it is overwritten without warning, allowing you, for example, to repeatedly republish the same Product version during testing. If Spaces are enabled in a development Catalog and you republish a previously published Product version to a different Space, it is removed from the previous Space.  
  
A development Catalog behaves the same as any other Catalog with regard to requiring approval for staging and publishing actions, if the Catalog has been configured to require approval.  
  
Note: In a production Catalog, the following conditions apply:
    - You cannot publish a Product to the Catalog if there is already a Product in the Catalog with the same name and version. Instead, you must create a new version of the Product for publication; see [Creating a new version of your Product](#). If Spaces are enabled in a production Catalog, you cannot publish a Product with the same name and version to more than one Space in the Catalog.
    - If this new Product contains one or more modified APIs, you **must** create new versions of the APIs for inclusion in the Product; see [Creating a new version of an API definition](#). If the Product contains a modified API and there is already a published API of the same name and version, the changes will **not** be published.
  - b. To enable the partition of this Catalog into Spaces, set the Spaces slider control to the On position, then click Confirm.  
For more information, see [Using syndication in IBM API Connect](#).
  - c. If the Catalog is the default staging Catalog, move the Default slider control to the On position, then click Confirm. Then, calls to APIs that are published to the Catalog can use a shorter URL that does not include the Catalog name.  
Note: You can enable Default only for one Catalog. If another Catalog is already set as the default Catalog you will be unable to enable the setting on this Catalog without first disabling the setting on the other Catalog.
7. To transfer ownership of the Catalog to a different user, click Overview, then proceed with the following steps:
  - a. Click Edit.
  - b. From the Select owner user drop-down menu, select the user who will become the new owner.
  - c. If Spaces are enabled in the Catalog then to also have the ownership of all Spaces in the Catalog transferred to the new Catalog owner, select Also transfer ownership of owned Spaces.
  - d. Click Save when done.Note:
  - To have permission to change the Catalog owner, you must be either the Catalog owner, or the owner of the provider organization that contains the Catalog.
  - When the ownership changes, the new owner is assigned administration privileges for the Catalog, and the previous owner has their administration privileges removed. If required, you can restore the privileges to the previous owner as described in [Managing Catalog membership](#).
  - The ownership can be transferred only to a user who is already a member of the Catalog. To transfer to a new user or another existing user, that user must previously have been added or invited to the Catalog, as described in [Managing Catalog membership](#).
8. To configure the gateway service settings for the Catalog, click Gateway Services, then proceed with the following steps:  
Note: If Spaces are enabled in the Catalog then you configure the gateway settings for the Spaces, **not** for the Catalog as a whole.
  - a. Click Edit.
  - b. Select the gateway services that you want to use with this Catalog.



Gateway services are configured by using the Cloud Manager UI. For details, see [Registering a gateway service](#). If any Catalog default gateway services have been configured, they will already be selected when you create a new Catalog; see [Configuring default gateway services for Catalogs](#). The TYPE column indicates the gateway type for each gateway service listed, DataPower® Gateway (v5 compatible), DataPower API Gateway, or Event Gateway Service if you have Event Endpoint Management enabled in Cloud Pak for Integration. For more information, see [API Connect gateway types](#).

c. Click Save to save your changes.

You publish APIs by adding them to a Product and then publishing the Product to a Catalog. To be able to publish Products to a Catalog, the Catalog must be assigned at least one gateway service so that the APIs in the Product are available to be called at a gateway service endpoint. You can assign more than one gateway service to a Catalog, and they can be of mixed types: DataPower API Gateway, DataPower Gateway (v5 compatible), and Event Gateway Service.

A Product is gateway type specific: DataPower API Gateway, DataPower Gateway (v5 compatible), or Event Gateway Service. By default, when you publish a Product to a Catalog, it is published to all the assigned gateway services of the same gateway type as the Product, but you can choose to publish the Product only to selected gateway services, of that type, at publish time; see [Publishing a draft Product](#) for more information. The APIs in the Product will be available to be called at all the specified gateway service endpoints.

If you select two or more Gateway services then, for analytics data to be available in the Developer Portal for this Catalog, the Gateway services must all be associated with the **same** analytics service. If the set of associated analytics services for the selected gateway services includes two or more different analytics services then the Developer Portal does **not** display analytics data. An analytics service is associated with a gateway service in the Cloud Manager user interface; see [Associating an analytics service with a gateway service](#).

Note: You can also configure the gateway service settings for the Catalog by using the developer toolkit CLI; for details, see [apic configured-gateway-services](#).

9. To specify the Product lifecycle state changes for which you want to enforce approval, complete the following steps:

a. Click Lifecycle Approvals in the Catalog settings navigation pane, then click Edit.

b. Select the required state changes, then click Save when done.

For example, if you select Publish and leave the other check boxes cleared, approval is required when anyone attempts to publish a Product, but no approval is required for any of the other lifecycle state changes.

Note: Approval for Product lifecycle state changes in a Catalog is disabled by default. You must explicitly enable the Product lifecycle state changes that you want to enforce.

c. To allow the originator of a Product lifecycle change to approve it themselves, move the Task self approval slider control to the On position.

Note: In a development Catalog, the originator of a Product lifecycle change can **always** approve it themselves. In a production Catalog, the originator of a Product lifecycle change can approve it themselves only if the Catalog setting Task self approval is enabled.

10. Click Save.

11. To configure the publish validations that are performed when a Product is published, complete the following steps:

a. Click Publish validations in the Catalog settings navigation pane.

b. Click Edit, and select or clear the validation check boxes as required.

The following validation options can be selected:


- Publishing APIs with empty paths is not allowed.
- Validate custom policy schema in assembly.
- Validate the references inside the API and perform additional OpenAPI validations.
- Publishing APIs with duplicate base paths is not allowed.

12. Click Save.

13. To configure the permissions that each role in API Manager has, complete the following steps:

a. Click Roles in the Catalog settings navigation pane.

To see the current permissions for a role, expand the role.

b. Click the options icon  alongside the role that you want to work with, then click Edit.

c. Select or clear the permissions check boxes as required.

To ensure a role can manage user-defined policies, select Manage for the Settings permission.

To grant a role the permission to approve a specific lifecycle state change, select the state change in the Product-Approval section.


Note: You cannot remove a permission that has been assigned to the role at the provider organization level. However, you can add permissions that haven't been assigned at the provider organization level.

d. Click Save when done.

14. To configure the default role permissions for consumer organizations that are created in this Catalog, complete the following steps:

a. Click Role Defaults in the Catalog settings navigation pane.

To see the current default permissions for a role, expand the role.

b. Click the options icon  alongside the role that you want to work with, then click Edit.

c. Select or clear the permissions check boxes as required.

d. Click Save when done.

e. To create a new role and assign default permissions to it, click Add, name the role and assign permissions, then click Save.


15. To manage the onboarding of API consumers into this Catalog, complete the following steps:

a. Click Onboarding in the Catalog settings navigation pane.

b. To select the registries that are used to authenticate users of the Developer Portal associated with this Catalog, click Edit in the Catalog User Registries section, select the required registries, then click Save.

For details on how to configure a user registry, see [Authenticating by using your enterprise user registry](#).

Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.

c. To set a default registry, click the options icon  alongside the required registry, then select Set default.

When an API consumer signs up to the Developer Portal, the specified registry is used by default, with the option to select another registry.

d. To allow API consumers to complete their own sign-up process to the Developer Portal, move the Self service onboarding slider control to the On position. If this option is disabled, an API consumer cannot sign-up without an invitation from a consumer organization owner.

e. To require approval for all new self service onboarding to the Developer Portal, move the Self service onboarding approval slider control to the On position. If this option is enabled, then any attempt by an API consumer to sign-up to the Developer Portal results in an approval request being sent. This request is displayed in the Task tab in the Catalog for the associated Developer Portal, from where the request can be approved or declined.

Note that Self service onboarding approval is honored whenever a consumer organization is created. So, even if an API consumer has already been approved to access a Developer Portal with a specific consumer organization, if they then try to create another consumer organization in the same Developer Portal,

their request will still go through the approval process.

Note: To enable self-service onboarding approval, you must also complete the following tasks:

- Configure the user registry to require the user's email address.
  - Configure the OIDC apps to send the user's email address for approval.
- f. To configure the ability for API consumers to invite collaborators and assign them to roles, click Edit in the Consumer invitation and roles section, select the options required, and click Save.
  - g. To configure the timeout for activation links that are sent in email invitations to application developers, click Edit in the Invitation Timeout section, specify the timeout length, then click Save.
  - h. To override the timeouts set by consumer organization invitations, enable the Override consumer organization's invitation timeout with catalog invitation timeout setting by moving the slider to the On position.  
The catalog's own timeout setting will be used instead.

16. To specify the user registries that are used to secure access to APIs in this Catalog, and to add the user registry to the Sandbox Catalog, complete the following steps:

- a. Click the Catalog settings tab, and then select API User Registries.
- b. Click Edit and select the user registries you want to use.  
For details on how to configure a user registry, see [Authenticating by using your enterprise user registry](#).
- c. Click Save when done.

Note:

- Only LDAP and Authentication URL registries can be used to secure access to APIs; therefore, only registries of these types are listed and available for selection.
- If Spaces are enabled in the Catalog then you specify the user registries for the Spaces, **not** for the Catalog as a whole. If you specify a user registry in one Space in a Catalog, it is deployed to the gateway services across all Spaces in that Catalog.  
For more information about enabling and managing Spaces, see [Using syndication in API Connect](#)

17. To specify the OAuth Providers that can be used to secure access to APIs in this Catalog, complete the following steps:

Note:

- If you want to specify an OAuth provider for a Catalog, ensure that the Sandbox Catalog is configured to use the same OAuth provider
- Any resources that the OAuth provider references, such as API user registries or TLS client profiles, must also be enabled in the Catalog
- If Spaces are enabled in the Catalog then you specify the OAuth providers for the Spaces, **not** for the Catalog as a whole. If you specify an OAuth provider in one Space in a Catalog, it is deployed to the gateway services across all Spaces in that Catalog.  
For more information about enabling and managing Spaces, see [Using syndication in API Connect](#).

- a. Click OAuth Providers in the Catalog settings navigation pane.
- b. Click Edit, then select the OAuth providers that you want to use.  
To configure an OAuth provider, see either [Configuring a native OAuth provider](#) using Cloud Manager or [Configuring a native OAuth provider](#) using API Manager.
- c. Click Save when done.

18. Optional: Configure vanity API endpoints for your Catalog.

For any API, there are two possible endpoints to consider:

- The gateway endpoint at which the API is invoked.
- The endpoint that is visible to the consumer in the Developer Portal.

If you do not configure vanity endpoints, these two endpoints are the same, and point to the gateway endpoint at which an API is invoked.

To have an endpoint visible to the consumer that is different to the gateway endpoint, you configure a vanity endpoint. A vanity endpoint represents the endpoint by which an API is known externally; that is, the endpoint that is published to the Developer Portal and is used by an application developer to invoke the API. The way in which the gateway endpoints for an API are determined depends on the OpenAPI version of the API, as follows:

**(OpenAPI 2.0):** For any enforced API in API Connect, the gateway endpoint formats for the API are as follows:

- If the OpenAPI definition has no **host** field, the API endpoint has the following format:

```
https://gateway_service_host/provider_organization/catalog/basepath
```

- If the OpenAPI definition has a **host** field (for example, **petstore.com**), the host is appended to the path after the **provider\_organization/catalog** segments, and the API endpoint has the following format:

```
https://gateway_service_host/provider_organization/catalog/host_field_value/basepath
```

Note: This is a change in behavior from IBM API Connect Version 5.0, where the **host** field is not included in the API endpoint.

In these endpoint URLs, the variables are as follows:

- **gateway\_service\_host** is the host name of the gateway service on which the API is running.
- **provider\_organization** is the value of the **name** field of the provider organization that contains the Catalog in which the API is published.
- **catalog** is the value of the **name** field of the Catalog in which the API is published.
- **basepath** is the value of the **basepath** field in the OpenAPI definition for the API.
- **host\_field\_value** is the value of the host field in the OpenAPI definition for the API.

**API Gateway only** **(OpenAPI 3.0):** If the API is enforced by the DataPower API Gateway, the value specified for the first **url** entry in the **servers** array in the OpenAPI definition of the API is used to determine the **basepath** element of the API endpoint, as follows:

- a. If a relative URL is specified for the server URL, that value is used as-is for the **basepath**, and the API endpoint has the following format:

```
https://gateway_service_host/provider_organization/catalog/basepath
```

- b. If a full URL is specified for the server URL, the scheme (**http://** or **https://**) is ignored by the DataPower API Gateway, and the remaining host name and **basepath** are used to form the API endpoint as follows:

```
https://gateway_service_host/provider_organization/catalog/host_name/basepath
```

In these endpoint URLs, the variables are as follows:

- **gateway\_service\_host** is the host name of the gateway service on which the API is running.
- **provider\_organization** is the value of the **name** field of the provider organization that contains the Catalog in which the API is published.
- **catalog** is the value of the **name** field of the Catalog in which the API is published.
- **host\_name** is the value of the host name in the server URL.

- *basepath* is the value of the **basepath** derived from the server URL.

For details on configuring the server URL in an OpenAPI 3.0 API definition, see [Defining servers for an API](#), [Defining servers for a Path](#), and [Defining servers for an operation](#).

Note: You can change your Catalog settings to use the IBM API Connect Version 5.0 endpoint behavior, including support for a feature known as *host to catalog mapping*, whereby an API can be invoked without having to include the provider organization or Catalog in the URL path; see [Retaining Version 5 vanity endpoint behavior in a Catalog](#).

To configure vanity API endpoints, so that you can publish endpoints that are different to these gateway endpoints, complete the following steps:

- Click API Endpoints in the Catalog settings navigation pane. The Vanity API Endpoint page opens.
  - Click Edit.
  - To display the current API base endpoint settings, select Display vanity endpoint.
  - Select the required Preference setting. Choose one of the following settings:
    - **Catalog priority:** Any host defined in the OpenAPI definition of the API is ignored and all API endpoints always point to the endpoints that you define here. For example, suppose you define a vanity API endpoint, with Catalog priority preference, to be `https://prod.acme.com/`. Then the API endpoint will be `https://prod.acme.com/basepath`.  
The way in which the **basepath** is determined depends on the OpenAPI version of the API, as follows:
      - **(OpenAPI 2.0):** The **basepath** is taken from the **basepath** field in the OpenAPI definition.
      - **API Gateway only (OpenAPI 3.0):** The **basepath** is taken from the first **url** entry in the **servers** array in the OpenAPI definition, ignoring any scheme (`http://` or `https://`) and host name.
    - **API priority:** The host defined in the OpenAPI definition for the API takes precedence. For example, suppose you define a vanity API endpoint, with API priority preference, to be `https://api.acme.com/`. Then the following rules determine the endpoint that is used when an API is called:
      - If the OpenAPI definition has a host defined, `test.acme.com` for example, then that value is used to determine the API endpoint. For example: `https://test.acme.com/basepath`.
      - If the OpenAPI definition has no host defined, the API endpoint will be `https://api.acme.com/basepath`, as determined by the vanity API endpoint setting.
- The way in which any defined host is determined depends on the OpenAPI version of the API, as follows:
- **(OpenAPI 2.0):** The host is taken from the **host** field in the OpenAPI definition.
  - **API Gateway only (OpenAPI 3.0):** The host is taken from the host name in the first **url** entry in the **servers** array in the OpenAPI definition.
- Supply one or more endpoints, as follows:
    - If you selected Catalog priority, click Add to enter one or more endpoints URLs, and an optional summary for each. Any of the endpoint URLs can be used to invoke an API.
    - If you selected API priority, enter the endpoint URL, and an optional summary.

Note: After configuring your Catalog to support your vanity URL, you must configure your external network to map the vanity endpoints to the corresponding gateway endpoints. This typically includes the following:

- A DNS mapping to ensure that the vanity host resolves to the gateway.
- Additional URL routing for the API as required.

For example, if your gateway has a default domain name of `apigw.dc.zone.mycompany.com` and an IP address of `29.12.141.150`, then the generic DNS configuration might look like:

```
api.acme.com. CNAME apigw.dc.zone.mycompany.com.
```

or

```
api.acme.com. A 29.12.141.150
```

Consult your network administrator and the DNS provider's documentation to do this configuration.

- To configure the TLS client profiles that are used with this Catalog, complete the following steps:

Note: If Spaces are enabled in the Catalog then you configure the TLS client profiles for the Spaces, **not** for the Catalog as a whole. If you configure a TLS client profile in one Space in a Catalog, it is deployed to the gateway services across all Spaces in that Catalog.

For more information about enabling and managing Spaces, see [Using syndication in API Connect](#).

- Click TLS Client Profiles in the Catalog settings navigation pane.
- Click Edit, then select the TLS client profiles that you want to use with this Catalog.  
For details on how to create a TLS client profile, see [Creating a TLS client profile](#).
- Click Save when done.

- To configure the Developer Portal, complete the following steps:

- Click Portal in the Catalog settings navigation pane, then click Create.
- Select the portal service that you want to use with this Catalog.  
Portal services are configured by using the Cloud Manager UI. For details, see [Registering a portal service](#).
- Optionally, override the default generated portal URL.  
The URL must have the following format:

```
https://host_name.portal.com
```

where *host\_name.portal.com* must resolve to the IP address of the Developer Portal machine.

For example:

```
https://myhost.mydomain.com
```

Important:

- The URL must be unique across all Catalogs.
- Mixed-case names are supported for resource names, however, portal endpoint URLs can support only lower-case. If a provider organization is created with a mixed-case name, you must edit the default generated portal URL to make the provider organization name lower-case in the URL.


To implement sub paths into your site URL, see [Sub paths for Developer Portal sites](#).

- Select the portal service that you want to use with this Catalog.  
For more information on using the Developer Portal, see [Developer Portal: Socialize your APIs](#).
- Click Create to save your changes.

Note: The account profile for the owner of the Catalog must have an email address specified to receive the creation notification, otherwise the Developer Portal creation operation will fail. To check whether your account profile has an email address, and specify a value, complete the following steps:

- Click the User icon , and then select My Account.

- b. Check the contents of the Email field, and specify a value if necessary, then click Save.

Note: If you want to edit the portal service or URL, or delete the portal site, return to this portal settings page, click the options icon , and click Edit or Delete.

21. To define Catalog properties, complete the following steps:
  - a. Click Properties in the Catalog settings navigation pane, then click Create.
  - b. Enter the property Name and Value, then click Create.

The properties that you define can be referenced in any of the API definitions in this Catalog. It is also possible to define properties that are specific to an API definition; see [Setting API properties](#). For information on how to reference a property in an API definition, see [Variable references in API Connect](#).

Note:

- If you define a Catalog property of the same name as an API property, the API property takes precedence over the Catalog property.
- If you change the value of a Catalog property, any API that references that property must be republished for it use the new value.

22. Optional: To configure the visibility of Products that are published to a Catalog or a Space, see [Configuring Product visibility in a Catalog or a Space](#).

## What to do next

---

Import your own policies into the Catalog. For details, see [Importing a user-defined policy into a Catalog](#).

Note: When you import a policy, its implementation is imported into all Gateway devices associated with the Gateway service that hosts the Catalog. Additionally, API Connect modifies some object names and file locations to mark them with the appropriate Catalog name and policy version.

For more information about how to create and manage user-defined policies, see [User defined policies](#).

---

## Configuring Product visibility in a Catalog or a Space

How to configure the visibility of Products at the Catalog or Space level.

### Before you begin

---

To configure Product visibility, you must be assigned to a role that has the Settings\_>\_Manage permission for that Catalog or Space. For more information on assigning Catalog permissions to a role, see [Creating and configuring Catalogs](#).

Note: All references in this topic to a Catalog can also be applied to a Space in a Catalog, unless specified otherwise. For more information about Spaces, see [Using syndication in API Connect](#).

### About this task

---

When you create a draft Product, you typically define the visibility in the Product document; see [Specifying the visibility of your Product](#), and [Creating a draft Product](#) for details. If you don't define any visibility settings in the draft Product, then the visibility defaults to the following settings:

```
visibility: {
 view: {
 enabled: true,
 type: 'public',
 orgs: []
 },
 subscribe: {
 enabled: true,
 type: 'authenticated',
 orgs: []
 }
}
```

When the Product is published, the visibility that is defined in the draft Product can be overridden by using the Products view within a Catalog in the API Manager UI; see [Changing the availability of a Product](#) for more information.

From IBM API Connect 10.0.5.1, you can also define a Catalog or Space setting to control the visibility of Products that are published to that Catalog or Space, as described in the following instructions. When this Catalog or Space setting is enabled, it can be used for the following two purposes:

- The setting can ensure that any Product published to that Catalog or Space, cannot have a visibility setting that exceeds the visibility level set at the Catalog or Space level.
- The setting can also be used as the default visibility setting for all Products that are published to that Catalog or Space. To use the setting as the default visibility, you must ensure that the Product doesn't have any visibility set during the publish, either saved in the Product document, or set during the publish itself.

Product visibility is picked up in the following order of precedence: Product override - Product document - Catalog or Space setting - default setting.

Note: You can also modify the Catalog or Space visibility settings by using the API Connect REST APIs; see the [API Connect REST API documentation](#).

## Procedure

---

To modify the Catalog or Space visibility settings by using the developer toolkit CLI, complete the following steps.

1. Log in to your API Connect Management server as a member of a provider organization, by using the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

Where:

- *mgmt\_endpoint\_url* is the platform API endpoint URL.
- *user\_id* is the user ID that you want to log in with. The user ID must be a member of a provider organization. This is an ID that you could also use to log in to the API Manager user interface.
- *password* is the password that is associated with the supplied user ID.
- *identity\_provider* is the identity provider that is used to authenticate the supplied user ID.

For example:

```
apic login --server platform-api.myserver.com --username myuser --password mypassword --realm provider/myldap
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details on how to log in to your management server from the CLI, see [Logging in to the management server](#).

2. Define your visibility settings by creating a YAML file with the following structure:

```
allowed_product_visibility: {
 view: {
 enabled: true_or_false,
 type: 'visibility_type',
 org_urls: [],
 group_urls: []
 },
 subscribe: {
 enabled: false,
 type: 'authenticated'
 }
}
```

Where, in the `view` section of the `allowed_product_visibility` property:

- **enabled** - can be set to `true` or `false`. The default setting is `false`.  
If set to `true` then any Product that is published to that Catalog or Space can have only a visibility that is the same or more restrictive than is configured in the `type` setting. If no visibility is set in the Product document, then the Catalog or Space setting is used as the default for the publish. If you want to always use the Catalog or Space setting as the default, then you must ensure that a Product visibility is not set.  
Note: If you are creating draft Products, and no visibility is set, then the value from the `allowed_product_visibility` field in the Sandbox Catalog is used as the default.  
If set to `false`, the feature is disabled, and the Product visibility settings that are defined in the Product document are used.
- **type** - if **enabled** is set to `true`, then this property specifies the maximum visibility of the Products that are published to this Catalog or Space. Valid settings are:
  - **public** - the Products are visible to all users.
  - **authenticated** - the Products are visible only to users who have successfully authenticated.
  - **custom** - the Products are visible only to the consumer organizations and consumer organization groups that are specified in `org_urls` and `group_urls`.
- **org\_urls** - if **type** is set to `custom`, then this property specifies the URLs of the consumer organizations that you want the Products to be visible to. Otherwise, leave empty.
- **group\_urls** - if **type** is set to `custom`, then this property specifies the URLs of the consumer organization groups that you want the Products to be visible to. Otherwise, leave empty.

The `subscribe` section is blocked at the Catalog or Space level, and cannot be edited. Subscribability is defined by the Product visibility setting, and is always more restrictive than the `view` section. For more information, see [Specifying the visibility of your Product](#).

3. Modify your Catalog settings by using the following command:

```
apic catalog-settings:update --server mgmt_endpoint_url --catalog catalog_name --org organization_name filename
```

Where `filename` is the name of the YAML file that you created in step 2.

Note: If Spaces are enabled for the Catalog, then the visibility settings must be set at the Space level by using the `--space` option.

4. Confirm your updated Catalog settings by using the following command:

```
apic catalog-settings:get --server mgmt_endpoint_url --catalog catalog_name --org organization_name --output -
```

## Related tasks

- [Creating and configuring Catalogs](#)

## Configuring sub paths for Developer Portal sites

You can increase the specificity of an IBM® API Connect Developer Portal site URL by using sub paths.

By adding additional text to the URL of a Developer Portal site, you can create more site permutations for an organization. Developer Portal site URLs must be unique across all Catalogs.

Important:

- Sub paths are case-sensitive.
- The Developer Portal host name must contain only **lowercase** characters.

You can create a Developer Portal in the following format:

```
https://host_name.portal.com/sub_path_1/sub_path_2/.../sub_path_n
```

Where *sub\_path* can be any text that you want to implement to specify your site URL. There is no limit to the number of sub paths that you can have for the first site that you want to create.

However, you must not create overlaps in the naming of your URLs. A URL of one site must not be able to be confused with pages belonging to another site. For example, if you create the following site URL:

```
https://host_name.portal.com/bank/safe/dollar
```

Then, you cannot have the following site URL permutations:

```
https://host_name.portal.com/bank/safe
```

or

```
https://host_name.portal.com/bank/safe/dollar/cent
```

However, you can lengthen or shorten a site URL if you change a single sub path value, or the *host\_name.portal.com* value. For example, if you created the following site URL:

```
https://host_name.portal.com/bank/safe/dollar
```

Then, you can create the following site URL permutations:

```
https://host_name.portal.com/bank/euro
```

and

```
https://host_name.portal.com/bank/euro/vault/cent
```

Important: If you already have a site URL that ends in *sub\_path* texts, you cannot create a new site URL with the sub path text at the start.

For example, if you create the following site URL:

```
https://host_name.portal.com/bank
```

then, you cannot create the following site URL:

```
https://bank.host_name.portal.com
```

---

## Managing Catalog membership

Manage Catalog membership by adding new users to the Catalog and assigning roles to the users.


### Before you begin

To manage Catalog members in the API Manager, you must be assigned to a role that has the Member\_>\_Manage permission for that Catalog. For more information on assigning Catalog permissions to a role, see [Creating and configuring Catalogs](#).

### About this task

The Catalog owner can invite members of the provider organization work with the Catalog in the API Manager. The invited users are called "Catalog members" and are different from consumer users who are members of [consumer organizations](#).

### Procedure

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
  2. Click the Members tab.
  3. To add an existing user, click Add\_>\_Add member. To add a new user, click Add\_>\_Invite member.  
If you are using the Add member option, you can add any user who is already a member of another Catalog, or of a Space, in the provider organization, and is neither currently a member of this Catalog, nor the Catalog owner.
  4. Enter the email address of a new user, or search for, and select, an existing user.
  5. Select the roles that you want to assign to the user.  
For details of the roles and the default permissions assigned to them, see [API Connect user roles](#). For details on how to create your own roles, see [Creating custom roles](#).
- If Spaces are enabled in the Catalog, any role that you assign to the user at the Catalog level is assigned automatically to the user in all Spaces in that Catalog. Furthermore, if a user had originally been assigned a role only in a specific Space in a Catalog and you subsequently assign the user that role at the Catalog level, the Space specific role assignment is lost and the user now has that role in all Spaces in the catalog.
- Note: You can subsequently change the roles assigned to a user by selecting or clearing the appropriate check boxes alongside that user on the Members tab. If a user was originally added to the provider organization, rather than to the Catalog itself, the following conditions apply:
- Any role assigned to the user at the provider organization level is assigned automatically to the user in all Catalogs, and cannot be removed at the Catalog level.  
For details on adding a user to a provider organization, see [Adding provider organization users and assigning roles](#).
  - In the Catalog, the user has the permissions that are configured for the role at the Catalog level.
  - Any role that hasn't been assigned to the user at the provider organization level can be assigned to the user at the Catalog level.
6. If you are adding an existing user, click Add. If you are adding a new user, click Invite.

---

## Importing a user-defined policy into a Catalog

You can make your user-defined policy available to API developers by importing it into an IBM® API Connect Catalog, or a Space in a Catalog.

### About this task

---

For instructions, see [Defining, packaging, and publishing a catalog-scoped policy for the API Gateway](#).

---

## Importing an OpenAPI extension into a Catalog

You can extend the OpenAPI specification by adding either a JSON or YAML extension schema to an API. An extension is imported into a Catalog, or a Space in a Catalog, then added to the API schema.

### Before you begin

---

You must possess Catalog edit permissions to complete this task.

### About this task



---

This task describes how to import an OpenAPI extension into a Catalog by using the browser based API Manager user interface. Alternatively, you can use the developer toolkit CLI; see [Working with OpenAPI extensions](#).

For an example of an extension schema file, see [Referring to an extension in an API definition](#).

### Procedure

---

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
2. If Spaces are enabled in the Catalog, select the Space that you want to work with by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.
3. Click the Catalog settings tab if you are working in a Catalog, or the Space settings tab if you are working in a Space.
4. Click Gateway services.
5. Alongside the Gateway service that you want to work with, click the options icon  and then click View extensions.
6. Click Upload then upload the extension file as indicated, either by dragging and dropping the file, or by browsing and selecting it. The extension schema file will be either a JSON or YAML file.
7. Click Upload when done.

### Results

---

The user-defined extension is now imported into a Catalog or Space, and can be used when developing an API by using the API editor in either the API Designer or the API Manager user interface.

### What to do next

---

Add the OpenAPI extension to your APIs as required; see [Adding an OpenAPI extension to an API](#).

---

## Retaining Version 5 vanity endpoint behavior in a Catalog

The behavior of vanity endpoints in a Catalog has changed in API Connect Version 10 compared with Version 5. However, you can modify your Catalog settings to retain the Version 5 behavior.

### About this task

---

A vanity endpoint represents the endpoint by which an API is known externally; that is, the endpoint that is published to the Developer Portal and is used by an application developer to invoke the API.

You can modify your Catalog settings to define vanity endpoints that retain the API Connect Version 5 behavior.

You can modify the Catalog settings in either of the following ways:

- [Use the API Manager user interface](#).
- [Use the developer toolkit CLI](#).

You can also use the API Connect REST APIs; see the [API Connect REST API documentation](#).

### Procedure

---

- To modify the Catalog settings to retain Version 5 vanity endpoint behavior by using the API Manager user interface, complete the following steps:
  - Click API Endpoints in the Catalog settings navigation pane. The Vanity API Endpoint page opens.
  - Click Edit, then select v5 legacy behavior.
  - To use the API Connect URLs, select Use gateway URLs. To configure a vanity endpoint, select Display vanity endpoint then provide the following information:
    - Endpoint: The URL of your vanity endpoint.
    - Gateway Uri: The URL of the gateway service that is enabled in the Catalog and is associated with the vanity endpoint; select from the list provided.
    - Summary: An optional description of the vanity endpoint.
- To modify the Catalog settings to retain Version 5 vanity endpoint behavior by using the developer toolkit CLI, complete the following steps:
  - Log in to your API Connect Management server as a member of a provider organization, by using the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

where:

- `mgmt_endpoint_url` is the platform API endpoint URL.
- `user_id` is the user ID that you want to log in with. The user ID must be a member of a provider organization. This is an ID that you could also use to log in to the API Manager user interface.
- `password` is the password associated with the supplied user ID.
- `identity_provider` is the identity provider that is used to authenticate the supplied user ID.

For example:

```
apic login --server platform-api.myserver.com --username myuser --password mypassword --realm provider/myldap
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`. For full details on how to log in to your management server from the CLI, see [Logging in to the management server](#).

- Define the configuration for your vanity endpoint by creating a YAML file with the following structure:

```
legacy_endpoint_behavior: enabled
v5_endpoint_substitution_behavior:
 base_endpoints:
 - endpoint: vanity_endpoint_url
 description: description
 gateway_service_url: gateway_service_url
 unenforced_api_base_endpoint: unenforced_api_base_endpoint
```

where:

- the `legacy_endpoint_behavior: enabled` setting specifies that the Catalog retains API Connect Version 5 behavior. A Catalog can support either Version 5 behavior or Version 10 behavior but not both.
- `vanity_endpoint_url` is the URL of your vanity endpoint.  
Note: API Connect Version 5 supports only the host name in a vanity endpoint URL; a URL of the form `https://host_name/path` is not supported. Therefore the vanity endpoint URL that you supply here must contain only a host name; for example, `https://vanity.com`.
- `description` (optional) is a description of your vanity endpoint.
- `gateway_service_url` is the URL of the gateway service that is enabled in the Catalog and is associated with the vanity endpoint.  
Note: You can determine the gateway service URL by using the following command:

```
apic configured-gateway-services:list --scope catalog --catalog catalog_name --server mgmt_endpoint_url --org organization_name
```

where:

- `gateway_service_name` is the name of the gateway service.
- `catalog_name` is the name of the Catalog.
- `organization_name` is the name of the provider organization that contains the Catalog.

The command output lists all the gateway services that are enabled in the Catalog, together with their gateway service URLs.

- `unenforced_api_base_endpoint` (optional) is a custom API URL that specifies the URL for APIs that are deployed to a third party gateway.  
Note: `base_endpoints` is an array. You can configure multiple vanity endpoints by defining multiple `endpoint` entries.

- Modify your Catalog settings by using the following command:

```
apic catalog-settings:update --server mgmt_endpoint_url --catalog catalog_name --org organization_name filename
```

where `filename` is the name of the YAML file that you created in step 2.

- Confirm your updated Catalog settings by using the following command:

```
apic catalog-settings:get --server mgmt_endpoint_url --catalog catalog_name --org organization_name --output -
```

---

## Using syndication in API Connect



With the API Connect syndication feature, you can partition your Catalogs into *Spaces*. Each Space is used by a different API provider development team and has its own set of management capabilities relating specifically to the APIs that the associated team publishes to that Space, enabling each team to manage their APIs independently.

When you stage or publish an API to a Catalog that has Spaces enabled, you specify the Space within that Catalog that you want to stage or publish to. However, application developers that access the Developer Portal for the Catalog are unaware of the Space partitioning of the Catalog and see the APIs as a coordinated offering.

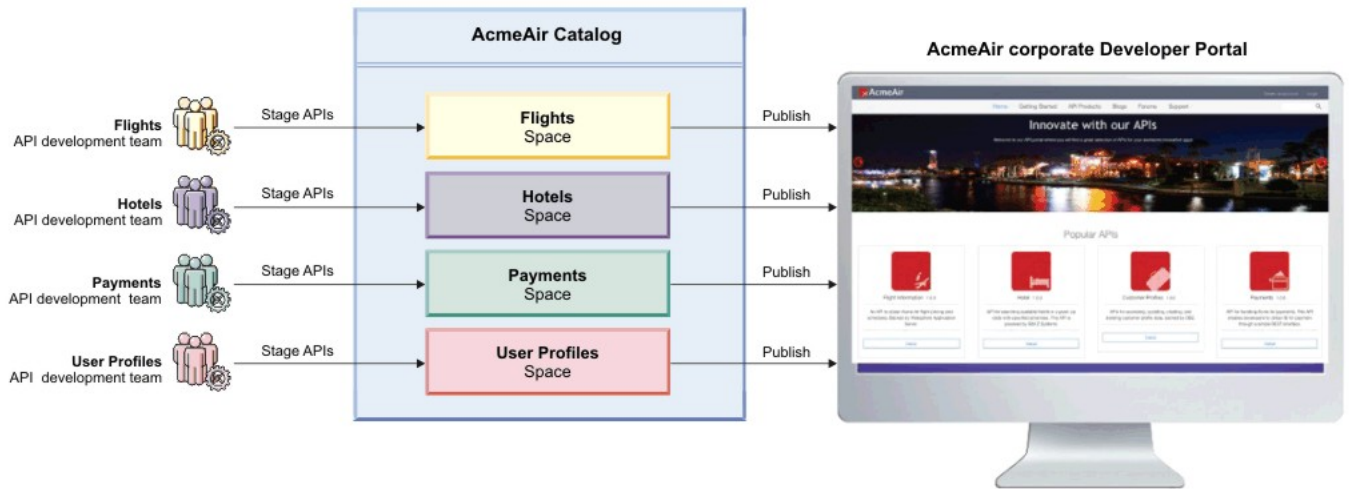
Each Space has its own Product lifecycle management, subscription approvals, and analytics data. You use Space specific access control to restrict user access to each Space; for example, a developer in the Flights team is able to stage APIs only to the Flights Space.

## Example

A travel company, AcmeAir, has separate API provider development teams for each of the following areas of their business:

- Flights
- Hotels
- Payments
- User Profiles

They have an AcmeAir corporate Catalog, with its associated corporate Developer Portal. They partition their Catalog into separate Spaces, one for each development team.



Spaces are described in detail in the following sections:

- [Enabling Spaces in a Catalog](#)  
To use the syndication feature in IBM API Connect, you must enable Spaces in any Catalog in which you require syndication capabilities.
- [Creating, modifying, and deleting Spaces](#)  
You use the API Manager user interface to create a new Space in a Catalog, modify the configuration, and owner of a Space, and delete a Space from a Catalog.
- [Working with Spaces](#)  
If Spaces are enabled in an IBM API Connect Catalog, you can select a specific Space to work with in the API Manager user interface. This enables you to manage the Products and APIs that are specific to that Space, and control user access to the Space.

## Enabling Spaces in a Catalog

To use the syndication feature in IBM® API Connect, you must enable Spaces in any Catalog in which you require syndication capabilities.

### Before you begin

If Spaces are enabled for a Catalog, Products (and their associated APIs) can be published only to a Space within that Catalog. For information about publishing, see [Staging a Product](#) for publishing by using the API Designer, and see [Publishing APIs](#) for publishing by using the developer toolkit.

Note: You cannot enable Spaces in a Catalog to which one or more Products have already been staged or published. If the Catalog does contain staged or published Products, first complete the following steps for each Product in the Catalog:

1. Remove all staged Products.
2. Retire, then remove, all published Products.

After enabling Spaces, re-stage or re-publish Products, and re-create application subscriptions, as required.


### About this task

By default, Spaces are disabled in a Catalog. You enable Spaces by modifying the Catalog settings. After Spaces are enabled for a Catalog, a default Space is automatically created that inherits all of the configuration settings from the Catalog.

Note: You can also enable Spaces by using the developer toolkit CLI; for details, see [apic spaces](#).

## Procedure

To enable Spaces in a Catalog, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
2. Click the Catalog settings tab.
3. Set the Spaces slider control to the On position.
4. In the Enable Spaces window, click Enable.

## Results

---

Spaces are enabled for your Catalog, and a default Space is created that automatically inherits all of the configuration settings from the Catalog. The Catalog owner is also the owner of the default Space.

## What to do next

---

You can modify Space settings, and create more Spaces; see [Creating, modifying, and deleting Spaces](#) and [Working with Spaces](#) for more information.

## Creating, modifying, and deleting Spaces

---

You use the API Manager user interface to create a new Space in a Catalog, modify the configuration, and owner of a Space, and delete a Space from a Catalog.

## Before you begin

---

Navigate to the Spaces page of the Catalog containing the Space you want to work with, by completing the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
2. Click the Spaces tab.

Note: You can also enable Spaces by using the developer toolkit CLI; for details, see [apic spaces](#).

## Procedure

---

- To create a new Space, complete the following steps:
    1. Click Add > Add a new Space.
    2. In the Space Details window, enter the Title of the Space and, optionally, a descriptive summary. A Name is entered automatically.  
Note: The value in the Name field is a single string that is used to identify the Space in developer toolkit CLI commands. The Title is used for display.  
To view the CLI commands to manage Spaces, see [apic spaces](#).
    3. Click Create to create the Space.

Note:


    - As the user who creates the Space, you are automatically the Space owner and have all Space permissions.
    - When you create a new Space, it does not automatically inherit all of the configuration settings from the Catalog. So you should check and modify the settings as necessary.
  - To modify the details of a Space, complete the following steps:
    1. On the Spaces page, select the Space whose details you want to modify, then click Space settings.
    2. Use the options in the navigation pane to modify the details as required.  
The following types of resource can be enabled in a Space:
      - Gateway services
      - API user registries
      - OAuth providers
      - TLS client profiles

Note:

      - If you enable a resource of any of the following types in one Space in a Catalog, it is deployed to the gateway services across all Spaces in that Catalog:
        - API user registries
        - OAuth providers
        - TLS client profiles
      - If you deploy any of the following types of resource to a gateway in a Space, it is actually deployed to the gateway at Catalog scope:
        - Custom policy
        - Global policy
        - Global policy pre-hook
        - Global policy post-hook
        - Gateway extension
    3. From the Select owner user drop-down menu, select the user who will become the new owner.
    4. Click Save when done.

Note:

    - To have permission to change the Space owner, you must be either the Catalog owner, the Space owner, or the owner of the provider organization that contains the Space.
    - When the ownership changes, the new owner is assigned administration privileges for the Space, and the previous owner has their administration privileges removed. If required, you can restore the privileges to the previous owner as described in [Managing Space membership](#).
    - The ownership can be transferred only to a user who is already a member of the Space or Catalog. To transfer to a new user or another existing user, that user must previously have been added or invited to the Space, as described in [Managing Space membership](#), or to the Catalog, as described in [Managing Catalog membership](#).
- To delete a Space, complete the following steps:

1. On the Spaces page, click the Manage icon  for the Space that you want to delete.
2. Click Delete, then click Confirm to confirm deletion.

Note:

- o You cannot delete a Space to which one or more Products have been staged or published.
- o If there is only one Space in a Catalog, you cannot delete it, but you can disable Spaces for the Catalog. However, you cannot disable Spaces in a Catalog to which one or more Products have already been staged or published. If any of the Spaces in the Catalog do contain staged or published Products, first complete the following steps for the Products in each of the Spaces:
  1. Remove all staged Products.
  2. Retire, then remove, all published Products.

Then, to disable Spaces for the Catalog, complete the following steps:

1. Click the Catalog settings tab in the Catalog.
2. Set the Spaces slider control to the Off position.

After disabling Spaces, re-stage or re-publish Products, and re-create application subscriptions, as required.

---

## Working with Spaces

If Spaces are enabled in an IBM® API Connect Catalog, you can select a specific Space to work with in the API Manager user interface. This enables you to manage the Products and APIs that are specific to that Space, and control user access to the Space.

---

### Procedure

To work with a Space, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
2. Click the Spaces tab.
3. Select the Space that you want to work with.

---

### What to do next

For details of the Space management tasks, see the following subtopics:

- [Managing Products in a Space](#)  
If Spaces are enabled in an IBM API Connect Catalog, you can use the API Manager user interface to separately manage the Products that are staged or published to each Space.
- [Managing subscription requests in a Space](#)  
If Spaces are enabled in an IBM API Connect Catalog, you can approve, or deny, individual requests to subscribe to Plans in Products that are published to the Space.
- [Managing application subscriptions in a Space](#)  
If Spaces are enabled in an IBM API Connect Catalog, you can use the API Manager user interface to separately manage the application subscriptions to Plans in the Products that are published to a Space.
- [Managing Space membership](#)  
If Spaces are enabled in an IBM API Connect Catalog, you can manage the members within the Space. You manage Space membership by adding new users to the Space and assigning roles to the users.
- [Managing user access in a Space](#)  
If Spaces are enabled in an IBM API Connect Catalog, you can manage the access that users have within the Space. You manage access by specifying the permissions that are assigned to user roles.
- [Managing Gateways in a Space](#)  
If Spaces are enabled in a Catalog, you can separately control which Gateway services are used by each of the Spaces in the IBM API Connect Catalog.

---

## Managing Products in a Space

If Spaces are enabled in an IBM® API Connect Catalog, you can use the API Manager user interface to separately manage the Products that are staged or published to each Space.

---

### Before you begin

To manage Products in a Space in the API Manager UI, a user must be assigned a role that has the Products\_>\_Manage permission. For more information on assigning Space permissions to a role, see [Managing user access in a Space](#).

---

### Procedure

To manage the Products that are staged or published to a specific Space, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
2. Click the Spaces tab.
3. Select the Space that you want to work with.

---

### What to do next

For details of the management tasks that you can perform on the Products in the Space, see [Working with Products](#).

---

## Managing subscription requests in a Space

If Spaces are enabled in an IBM® API Connect Catalog, you can approve, or deny, individual requests to subscribe to Plans in Products that are published to the Space.

### Before you begin


---

To manage Space subscription requests in the API Manager UI, a user must be assigned a role that has the Subscription Approvals, Manage permission. For more information on assigning Space permissions to a role, see [Managing user access in a Space](#).

### Procedure

---

To manage the subscription requests in a Space, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
2. Click the Spaces tab.
3. Select the Space that you want to work with.
4. Click the Tasks tab.  
The Tasks page lists only the subscription requests for the selected Space.

### What to do next

---

For details on how to manage approvals, see [Approving Product lifecycle and subscription requests](#).

---

## Managing application subscriptions in a Space

If Spaces are enabled in an IBM® API Connect Catalog, you can use the API Manager user interface to separately manage the application subscriptions to Plans in the Products that are published to a Space.



### Before you begin

---

To manage application subscriptions in the API Manager UI, a user must be assigned a role that has the Subscriptions, Manage permission. For more information on assigning Space permissions to a role, see [Managing user access in a Space](#).

### Procedure

---

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
2. Click the Spaces tab.
3. Select the Space that you want to work with.
4. In the navigation pane of the API Manager UI, click the Applications tab.  
All applications that have been registered in the Developer Portal associated with this Space are listed.
5. Click the options icon  alongside the application you want to work with, then click View Subscriptions.

### What to do next

---

For details of the management tasks that you can perform on application subscriptions in a Space, see [Working with application subscriptions](#).

---

## Managing Space membership

If Spaces are enabled in an IBM® API Connect Catalog, you can manage the members within the Space. You manage Space membership by adding new users to the Space and assigning roles to the users.

### Before you begin

---

To manage Space members in the API Manager UI, a user must be assigned a role that has the Child, Manage permission for the Catalog that contains the Space. For more information on assigning Space permissions to a role, see [Managing user access in a Space](#).

### About this task

---

Note: You can add the same user to two or more Spaces and assign different roles in each, allowing a user to have differing levels of access in different Spaces.

### Procedure

---

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
2. Click the Spaces tab.
3. Select the Space that you want to work with.

4. Click the Members tab.
5. To add an existing user, click [Add](#) > [Add member](#). To add a new user, click [Add](#) > [Invite member](#).  
If you are using the Add member option, you can add any user who is already a member of another Space, or of a Catalog, in the provider organization, and is neither currently a member of this Space, nor the Space owner.
6. Enter the email address of a new user, or search for, and select, an existing user.
7. Select the roles that you want to assign to the user.  
For details of the roles and the default permissions assigned to them, see [API Connect user roles](#). For details on how to create your own roles, see [Creating custom roles](#).

If a user was originally added either to the provider organization, or to the Catalog that contains the Space, rather than to the Space itself, the following conditions apply:

- Any role assigned to the user at the provider organization or Catalog level is assigned automatically to the user in all Spaces, and cannot be removed at the Space level.

For details on adding a user to a provider organization, see [Adding provider organization users and assigning roles](#).

For details on adding a user to a Catalog, see [Managing Catalog membership](#).

- In the Space, the user has the permissions that are configured for the role at the Space level.  
For details on configuring role permissions for a Space, see [Managing user access in a Space](#).

- Any role that hasn't been assigned to the user at the provider organization or Catalog level can be assigned at the Space level

Note: You can subsequently change the roles assigned to a user by selecting or clearing the appropriate check boxes alongside that user on the Members page.

8. If you are adding an existing user, click Add. If you are adding a new user, click Invite.

---

## Managing user access in a Space

If Spaces are enabled in an IBM® API Connect Catalog, you can manage the access that users have within the Space. You manage access by specifying the permissions that are assigned to user roles.



---

### Before you begin

To manage Space permissions in the API Manager UI, a user must be assigned a role that has the `Child_>Manage` permission for the Catalog that contains the Space. For information on adding users to a Space and assigning user roles, see [Managing Space membership](#).

---

### Procedure

1. In the navigation pane of the API Manager UI, click  [Manage](#), then select the Catalog that you want to work with.
2. Click the Spaces tab.
3. Select the Space that you want to work with.
4. Click the Space settings tab.
5. Click Roles, click the options icon  alongside the role whose permissions you want to modify, then click Edit.
6. Select the required permissions, then click Save.

---

### Related tasks

- [Creating custom roles](#)

---

### Related information

- [API Connect user roles](#)

---

## Managing Gateways in a Space

If Spaces are enabled in a Catalog, you can separately control which Gateway services are used by each of the Spaces in the IBM® API Connect Catalog.

---

### Before you begin

To manage Space Gateway settings in the API Manager UI, a user must be assigned a role that has the `Child_>Manage` permission for the Catalog that contains the Space. For information on adding users to a Space and assigning user roles, see [Managing Space membership](#).

---

### About this task

Note: If Spaces are enabled in a Catalog then you configure the Gateway settings for each Space separately, **not** for the Catalog as a whole.

---

### Procedure

To manage the Gateway settings for a Space, complete the following steps:

1. In the navigation pane of the API Manager UI, click  [Manage](#), then select the Catalog that you want to work with.

2. Click the Spaces tab.
3. Select the Space that you want to work with.
4. Click the Space settings tab.
5. Click Gateway services.
  - The Gateway services that are currently enabled for the Space are listed.
6. To change the Gateway service configuration for the Space, click Edit, then select which Gateway services you want the Space to use.
7. Click Save to save your changes.

## Related information

- [Configuring the initial Gateway service](#)

## Working with Products

In IBM API Connect, Plans and APIs are grouped together in Products.

Information about managing Products in the API Manager UI can be found in the following topics.

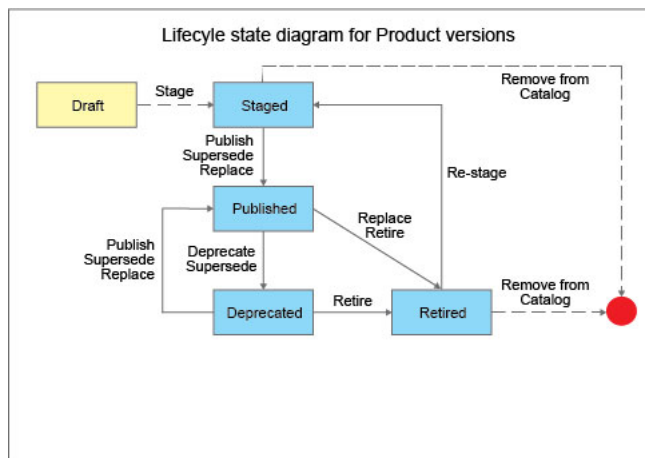
- **The Product lifecycle**  
When you manage your Product versions, you move them through a series of lifecycle states. From initially staging a Product version to a Catalog, through to publishing to make the Product version available to your application developers, and to eventual retiring and archiving. The syndication feature in IBM API Connect means that Product lifecycle states can also be managed within Spaces in the associated Catalog.
- **Monetizing your Products**  
In addition to offering free Plans for your customers to use your Products, you can also define Plans that automatically bill your customers who are using your Products in IBM API Connect.
- **Managing your Products**  
You can manage your Products in API Manager by using the Products page of the associated Catalog. In this view, you can move the Products through their lifecycle and control who can see or subscribe to the Products. The syndication feature in IBM API Connect means that Products can also be managed by using the Products tab of the associated Space.

## The Product lifecycle

When you manage your Product versions, you move them through a series of lifecycle states. From initially staging a Product version to a Catalog, through to publishing to make the Product version available to your application developers, and to eventual retiring and archiving. The syndication feature in IBM® API Connect means that Product lifecycle states can also be managed within Spaces in the associated Catalog.

## Product lifecycle state diagram

The following diagram shows the possible lifecycle states for a Product version, and the Product management operations that move a Product version from one lifecycle state to another. For example, the Retire operation moves a Product version from the Published to the Retired state.



Note: The same Product lifecycle states apply irrespective of whether your Product is managed within a Catalog, or within a Space in a Catalog. For more information about the syndication feature, see [Using syndication in API Connect](#).


If approval is required for a Product management operation, an approval request is sent and the Product version moves to the pending state. When the request is approved, the operation is completed and the Product version moves to the next lifecycle state. If approval is not required, the operation is completed immediately.

Note: Approval is not required for the following lifecycle state transitions:

- Retired to Staged.
- Deprecated to Published.

For information on configuring Product lifecycle approvals for a Catalog, see [Creating and configuring Catalogs](#). For information on approving requests, see [Approving Product lifecycle and subscription requests](#).

Note:

- Approval for Product lifecycle state changes in a Catalog is disabled by default. You must explicitly enable the Product lifecycle state changes that you want to enforce.
- Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
- You can view the history of Product lifecycle requests and approvals, by clicking the options icon  alongside the Product that you want to work with, and selecting View Approval History.

The following sections describe the various lifecycle states for a Product version.

Note: All references in this topic to a Catalog, can also be applied to a Space in a Catalog, unless specified otherwise.

## Draft

---

The draft state for a Product or API is when a Product or API definition is not deployed and is not associated with any Catalog.

## Staged

---

When you stage a Product, a copy of the Product version is deployed to the target Catalog. Staged is the initial state when you publish a Product. When a Product is in the staged state, it is not yet visible to, or subscribable by, any developers. For more information about staging a Product, see [Staging a Product](#).

You stage a Product so that the appropriate approvals, internally within the organization, can be given for it to then be published. For information on configuring Product lifecycle approvals for a Catalog, see [Creating and configuring Catalogs](#). For information on approving requests, see [Approving Product lifecycle and subscription requests](#). For information on publishing a Product, see [Publishing a Product](#).

## Published

---

When you publish a Product, a fixed copy of the Product version is deployed to the target Catalog. The Product version is visible to, and subscribable by, the targeted developers or communities. When a Product is published in a Catalog, the visibility and subscription settings can be changed for the published version of that Product. Any further changes require a new version of the Product to be staged and published before they take effect.

If you replace a Published Product with a Staged or Deprecated Product, the replacement Product is published, and the replaced Product is retired.

If you supersede a Published Product with a Staged or Deprecated Product, the superseding Product is published, and the superseded Product is deprecated.

For more information about publishing a Product, see [Publishing a Product](#).

Note: If you want to update a Product, you must republish it. The `products:update` command only updates a Product's metadata.

## Deprecated

---

When you deprecate a Product, the Product version is visible only to developers whose applications are currently subscribed. No new subscriptions to the Plans in the Product are possible. For more information about deprecating Products, see [Deprecating a Product](#).

A Product is also deprecated if you supersede it with another Product. For more information, see [Superseding a Product with another Product](#).

## Retired

---

When you retire a Product, the Product version can neither be viewed nor can its Plans be subscribed to, and all of the associated APIs are taken offline. For more information about retiring Products, see [Retiring a Product](#).

A Product is also retired if you replace it with another Product. For more information, see [Replacing a Product with another Product](#).

## Related concepts

---

- [Working with Catalogs](#)

## Related tasks

---

- [Managing your Products](#)
- [Working with Spaces](#)

## Monetizing your Products

In addition to offering free Plans for your customers to use your Products, you can also define Plans that automatically bill your customers who are using your Products in IBM® API Connect.

API Connect includes a subscription billing feature that allows API providers to define pricing Plans in their API Products, and monetize their API offerings. If a Product contains a pricing Plan, API consumers must enter their payment information into the Developer Portal before they can subscribe to that Plan.

API Connect supports integration with Stripe Subscription Billing, an independent cloud service that manages monetized product Plans, customers, their payment information, and their subscription history, in order to generate monthly invoices and charge customers automatically. With this integration, Stripe serves as both the subscription billing system and the payment processing system.

You can define both free Plans and paid Plans within the same Product. To charge your customers more for greater access to your APIs, you can create multiple Plans with different tiers of prices and rate-limits. For example, an introductory Plan might have a low monthly price and restrictive rate-limiting, while a higher tier Plan might couple a higher monthly price with less restrictive rate limiting controls. Your customers can then select the Plan that is appropriate for their usage rates. As your customers have different needs for the use of your Products, so you can offer different levels of service. For example, a smaller customer might need a rate of only 5 API calls per hour,

whereas a larger company might need a rate of 1000 API calls per hour. You can create the Plans that are appropriate for your customers, and they can subscribe to the service and pricing Plan that is appropriate for them.

Restriction: The following limitations apply to the subscription billing feature:

- Billing and monetization requirements vary by country, where IBM and the independent cloud billing provider, Stripe, cannot guarantee every country requirement is supported. Country requirements change, and are out of scope for this document. Some limitations on storing and passing fields to the Stripe billing provider might exist, when fields are unique to a subset of countries. For example, the passing of a billing address from API Connect to Stripe, which is required by some EU countries due to GDPR and taxation laws, is not currently supported.
- Two data center disaster recovery configurations including the optional billing microservice are not supported at this time.
- After a configured billing resource is in use by a Catalog, it might not be possible to remove it. Do not add your Stripe test keys to production Catalogs. If you want to prototype with your non-production Stripe keys, create a separate temporary Catalog.
- If an API consumer deletes their payment method, or their credit card expires, or their card is out of credit after they have initially subscribed and prepaid for their first month, there is no mechanism to alert the provider organization or suspend API access automatically in response. Provider organizations can monitor their Stripe dashboards for non-payment events and, if necessary, notify the consumer, suspend the application for that subscription, or cancel the API Connect subscription manually. The URL of the API Connect subscription is saved in the metadata of the Stripe subscription object, to assist in reverse lookups.
- Suspending an application does not automatically pause the billing cycles of the subscriptions for that application.
- You can republish a Product that includes billing only when the Catalog Production mode is set to On.

Though much of the procedure is the same for setting up a paid Plan and a free Plan, there are some extra actions that are needed before you can start defining paid Plans. See the following topics for more information:

1. First, you must configure a billing integration resource for your provider organization; see [Adding a billing integration resource](#).
2. Before you can publish a Product Plan with pricing to a Catalog, you must add your billing integration resource to that Catalog; see [Adding a billing integration resource to a Catalog](#).
3. Also before you can publish to a Catalog, an administrator must enable the Stripe payment method module in the Developer Portal for that Catalog; see [Configuring Stripe in the Developer Portal](#).
4. After the Catalog and Developer Portal are configured, you can create Products that include pricing Plans; see [Defining a Product with billing integration](#).

Note:

- Before you can create billing resources in provider organizations, the billing microservice must be enabled on your management system by your system administrator. See [Configuring monetization on VMware](#) and [Configuring monetization on Kubernetes](#) for more information.
- You can also create and manage billing integration resources by using the developer toolkit CLI. For more information, see [Command-line tool reference for the developer toolkit](#).

---

## Related tasks

- [Publishing a Product](#)
- [Managing your Products](#)

---

## Related reference

- [Troubleshooting your billing configuration](#)

---

# Adding a billing integration resource

To enable the monetization of your Product Plans, you must add a billing integration resource in your IBM® API Connect provider organization that defines the configuration data needed to synchronize with an external subscription billing system.

---

## Before you begin

To use Stripe as your credit card processing vendor, you must have port 443 open to HTTPS communication between the Stripe API and your Developer Portal management and the Management cluster servers. See [Firewall requirements on Kubernetes](#) and [Firewall requirements on VMware](#) for more information about this requirement.

Before you can create billing resources in provider organizations, the billing microservice must be enabled on your management system by your system administrator. See [Configuring monetization on VMware](#) and [Configuring monetization on Kubernetes](#) for more information.

You must either be the provider organization owner, or have `Settings: Manage permissions`, to complete this task.

You must have an account with Stripe to be able to complete this task. If you do not already have a Stripe account, you can set one up at: [www.stripe.com](http://www.stripe.com).

---

## About this task

API Connect supports integration with Stripe Subscription Billing, an independent cloud service that manages monetized product Plans, customers, their payment information, and their subscription history, in order to generate monthly invoices and charge customers automatically. With this integration, Stripe serves as both the subscription billing system and the payment processing system. When API consumers subscribe to Product Plans with billing in the Developer Portal, they set up automatic payments through a credit card registered with Stripe. To enable this billing process, you must first specify your Stripe account information by creating a billing integration resource. The subscription payment amount is processed from the account that is provided by the API consumer, and credited to the account that you provide as the API provider.


Important: Each Stripe account comes with two sets of API keys, one for testing, and one for production. Each set of API keys has a distinct namespace for Stripe objects. Test API keys cannot see objects created by production API keys, and vice versa. You cannot switch the API keys of one of your billing integrations with the keys from another account, or swap your test and production keys over, as that would prevent API Connect from resolving the Stripe objects that were created by using the old keys. If you want to use your Stripe test keys, you must create a separate Catalog for testing, and not add your Stripe test keys to production Catalogs.



## Procedure

---

To create a billing integration resource, complete the following steps:

1. In the API Manager, click  Resources.
2. Ensure that you are in the provider organization to which you want to add the billing integration.
3. Select Billing, and then click Add.  
The Add billing integration page for Stripe integration is displayed.
4. Enter a Title for your billing integration.
5. The Name is auto-generated based on the Title that you enter, and is a single string that can be used in developer toolkit CLI commands.
6. Enter the Publishable key and Secret key for your Stripe account.  
Refer to your Stripe dashboard to get your publishable key and secret key; see <https://dashboard.stripe.com/apikeys>.
7. Click Add to create your billing integration resource.

## Results

---

Your new billing integration resource is listed in the table on the Billing dashboard. The table also displays the state of the job queue for the billing integration. Any issues with the job queue are displayed in this table. See [Billing integration resource job queues](#) for more information.

## What to do next

---

To be able to publish monetized Product Plans, you must add the billing integration resource to your Catalog, and to the Sandbox Catalog. See [Adding a billing integration resource to a Catalog](#) for details.

## Related tasks

---

- [Defining a Product with billing integration](#)

## Related reference

---

- [Troubleshooting your billing configuration](#)



---

## Billing integration resource job queues

Billing integration resources use job queues to manage the bulk Product lifecycle operations in IBM® API Connect.

Subscription billing systems such as Stripe, typically provide APIs that support updating a single resource at a time. However, IBM API Connect has Product lifecycle operations that can modify many resources in a single transaction, such as migrating a set of subscriptions from one Product to another Product. In order to bridge the mismatch between these two APIs, API Connect adds a job queue to each billing integration resource in your provider organization. Jobs on this queue are responsible for mapping bulk operations to a series of API calls against the subscription billing system. Jobs run one at a time, in the order that they were created. When a job completes it's deleted from the queue, and the next job runs. When all of the jobs in the queue complete running and the queue is empty, the data model of the subscription billing system should be up-to-date with the changes made in API Connect.

There are potential issues when moving this type of synchronization to background jobs. Network connectivity, outages at the billing provider, or rolling of the API keys of the billing system, could all cause the job queue to be blocked if the root issue does not resolve by itself in the span of an hour. If a job encounters an error, such as a network connectivity error, the job is retried up to five times over the course of 70 minutes, before changing the state of the job from Running to either Blocked or Failed. Only when the state of the head job is blocked or failed, will the Retry Blocked Jobs action appear in the options menu. So if an error is shown but the job is still attempting to run, you need to wait up to 70 minutes before the Retry Blocked Jobs action appears.

The job queue status can be viewed in the table on the Billing dashboard. The table lists any blocked jobs, and the error that is blocking them. If the error indicates that your API keys are no longer valid, you must update your billing resource to contain the correct values. After the root cause is resolved, you can click the  options menu on the row of the billing integration resource, and click Retry Blocked Jobs. This action retries the jobs on the job queue. After a few minutes the number of jobs should go down. Note that you can click the Refresh icon  to refresh the results from the server.

## Related concepts

---

- [Monetizing your Products](#)

---

## Adding a billing integration resource to a Catalog

To be able to publish monetized Product Plans, you must add a billing integration resource to your Catalogs.

## Before you begin

---

You must have created a billing integration resource. See [Adding a billing integration resource](#) for details.

You must either be the provider organization owner, or have Settings: Manage permissions, to complete this task.

## About this task

---

Before you can publish any Products that include pricing Plans, you must add the billing integration resource to the Catalog where you will create the monetized Products. When API consumers subscribe to Product Plans with billing in the Developer Portal, they set up automatic payments through a credit card registered with Stripe. To enable this billing process, after you have specified your Stripe account in a billing integration resource, you must then apply that resource to your Catalogs. The subscription payment amount is processed from the account that is provided by the API consumer, and credited to the account that is provided in the billing integration resource.


Note:

- Before you can publish a monetized Product Plan to a Catalog, the Stripe payment method module in the Developer Portal must also be enabled; see [Configuring Stripe in the Developer Portal](#).
- Each Stripe account comes with two sets of API keys, one for testing, and one for production. Each set of API keys has a distinct namespace for Stripe objects. Test API keys cannot see objects created by production API keys, and vice versa. You cannot switch the API keys of one of your billing integrations with the keys from another account, or swap your test and production keys over, as that would prevent API Connect from resolving the Stripe objects that were created by using the old keys. If you want to use your Stripe test keys, you must create a separate Catalog for testing, and not add your Stripe test keys to production Catalogs.

## Procedure

---

To add a billing integration resource to a Catalog, complete the following steps:

1. In the API Manager, click  Manage.
2. Select the Catalog tile on the Manage page that you want to update, then click the Catalog settings tab.
3. Select Billing, and then click Edit.
4. Select the billing integration resource that you want to apply to the Catalog, and click Save. You should apply only one billing integration resource to each production Catalog.

The billing integration resource is now listed in the Billing section.

Note: More than one billing resource can be applied to the Sandbox Catalog.

## Results

---

The billing integration resource is added to the Catalog.

## What to do next

---

The Stripe payment method module must be enabled in the Developer Portal for that Catalog; see [Configuring Stripe in the Developer Portal](#) for details.

After both the Catalog and the Developer Portal are configured, you can create Products with pricing Plans. See [Defining a Product with billing integration](#) for more information.

## Related concepts

---

- [Considerations when changing a Product lifecycle with billing integration](#)

## Related tasks

---

- [Defining a Product with billing integration](#)

## Related reference

---

- [Troubleshooting your billing configuration](#)

## Defining a Product with billing integration

---

To monetize your APIs, you must create Product Plans with pricing information that allow API consumers to subscribe to your Plans with integrated billing.

## Before you begin

---

You must have a billing integration resource, and this resource must be added to the Catalog that you want to publish your Product to, as well as to the Sandbox Catalog. See [Adding a billing integration resource](#), and [Adding a billing integration resource to a Catalog](#) for details.

Also, the Stripe payment method module must be enabled in the Developer Portal for the Catalog; see [Configuring Stripe in the Developer Portal](#) for details.

## About this task

---

When you decide to charge customers to use your API Products, your customers have different requirements for the number of API calls that they are willing to pay for. You can solve this issue by defining Plans at different levels of service, and at different costs.

Within your Product, you can define free Plans as well as Plans that contain billing integration. The following instructions guide you through defining a Plan with pricing information. For more detailed information about creating Products, see [Creating a draft Product](#) and [Editing a draft Product](#).


Important: Products that contain a billing integration cannot be re-staged into non-production Catalogs. To experiment with billing on the Sandbox Catalog, or any other non-production Catalogs, you must publish new versions of your monetized Product. This practice aligns with how you have to work with production Catalogs currently, as

they also don't support the re-staging of existing versions of the same Product.

## Procedure


---

To add pricing information to a Product Plan, complete the following steps:

1. In the API Manager, click  Develop.
2. Select the Products tab.
3. Select the Product that you want to add billing integration to, or create a new draft Product by clicking Add > Product and following the wizard.
4. With the Product open and the Design tab selected, select Plans in the navigation menu.
5. In the Billing integration section, select the billing integration resource that you want to apply to the Product.  
In the Plans section, you can then either edit the Default Plan to add pricing information, or create a new Plan with pricing information.
6. If this is a new Plan, enter a Title, and if you require approval for someone to subscribe to this Plan also select the Approval checkbox.
7. In the Plan pricing section, complete the following steps to add the pricing information:
  - a. Change the toggle to On for Plan pricing.  
The Plan pricing definition section is displayed.
  - b. Optional: If you want to include any free trial days in your Plan, select Include free trial days, and then enter the number of trial days that a subscriber can use the Plan without charge, after which their billing cycle begins.
  - c. Select the Currency for the billing process use.
  - d. Enter the Price per month to bill the subscriber for.  
If the selected currency supports fractional units, enter the price including any fractional units, such as cents.
8. Optional: In the Plan rate limits section, set the rate limits by entering the number of calls that are allowed per unit of time.  
Remember that this rate limit applies only to this Plan. You can set multiple rate limits, at second, minute, hour, day, and week time intervals. You can override this Plan rate limit for specific endpoints if you prefer, adding a different rate limit or enabling unlimited calls to it. You can also create more Plans with different rate limits. For more information about rate limits, see [Understanding rate limits for APIs and Plans](#).
9. Optional: Select the Hard limit checkbox if you want the gateway to reject any calls beyond the specified rate, rather than adding them to a queue to be resolved when the limits reset.
10. Optional: Add more burst limits, GraphQL limits, and assembly limits, as appropriate for your APIs.
11. Complete the Plan APIs section if you want to customize the APIs and operations that are available in this Plan.
12. If you want to add more Plans to the Product, repeat the process from Step 6.
13. Click Save to save the Product.

## Results

---

Your Product with billing integration and a pricing Plan is listed in the Products dashboard of the Develop area. If your Product doesn't appear, click the Refresh icon  to refresh the results from the server.

## What to do next

---

Stage your Product to a Catalog. For more information, see [Staging a draft Product](#).

## Related concepts

---

- [Working with Products](#)
- [Considerations when changing a Product lifecycle with billing integration](#)

## Related reference

---

- [Troubleshooting your billing configuration](#)

## Managing your Products

---

You can manage your Products in API Manager by using the Products page of the associated Catalog. In this view, you can move the Products through their lifecycle and control who can see or subscribe to the Products. The syndication feature in IBM® API Connect means that Products can also be managed by using the Products tab of the associated Space.

## Before you begin

---

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Catalog that contains the Product. If you have View permission for Products, you have read-only access to the Product management page. For information on configuring Product management permissions for a Catalog, see [Creating and configuring Catalogs](#).

The syndication feature in IBM API Connect means that Products can be contained within a Space in a Catalog. In this case, to complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Space that contains the Product. For information on configuring Product management permissions for a Space, see [Managing user access in a Space](#).

## About this task

---





For more information about the Product lifecycle, see [The Product lifecycle](#).  
For more information about the syndication feature, see [Using syndication in API Connect](#).

For information about Product billing, see [Monetizing your Products](#).

## Procedure

---

To manage your Product, complete the following steps.

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.  
The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.
2. If the Product that you want to work with is contained within a Space, select the required Space by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.The Products page of the Space opens, and all of the Products available in that Space are displayed.
3. Expand a Product to see details of the APIs and Plans in that Product.  
  
Manage options are also available, depending on the state of the Product. To view the manage options, click the options icon  and select the options that you require.
4. To manage the lifecycle of a version of a Product, click the options icon  alongside the Product version, and select the required lifecycle action.
5. To set the migration target of a version of a Product, click the options icon  alongside the Product version, and select Set migration target. In the Set migration target window, select the Product that you want to set as the migration target, and select Next. Then map the migration source Plans to the migration target Plans, and click OK.  
Restriction: If the Plan is part of a Product that is contained within a Space in the Catalog, and the migration action is being made at the Space level, the Plan that you are migrating subscriptions from must be located in the same Space as the Plan you are migrating to. If the migration action is being made at the Catalog level, subscriptions can be migrated across Spaces. For more information about the Space feature, see [Using syndication in API Connect](#).

- **[Considerations when changing a Product lifecycle with billing integration](#)**

There are special considerations for lifecycle operations for Products that contain a billing integration in IBM API Connect.

- **[Publishing a Product](#)**

APIs become accessible when a Product is published and made visible on the IBM API Connect Developer Portal for use by application developers. A Product can be published to selected communities of application developer organizations, and the Plans within the Product can be used to tailor access and visibility further.

- **[Migrating application subscriptions to another Product](#)**

By using the Products view within a Catalog in API Manager, you can migrate application subscriptions between Products. The syndication feature in IBM API Connect means that you can also use the Products view within a Space in a Catalog to migrate application subscriptions.

- **[Migrating application subscribers to new Product versions](#)**

When new versions of Products are created in IBM API Connect, there are a number of ways that subscribers can be moved to the Plans of the new Product.

- **[Changing the availability of a Product](#)**

You can change the availability of a Product and the associated Plans by using the Products view within a Catalog in API Manager. The syndication feature in IBM API Connect means that you can also use the Products view within a Space in a Catalog to change the availability of a Product.

- **[Deprecating a Product](#)**

You can deprecate a published Product by using the Manage Products page within a Catalog in API Manager. When you deprecate a Product, application developers that are already subscribed to the Product can continue to use it, but no new developers can subscribe to the Product. The syndication feature in IBM API Connect means that you can also use the Manage Products page within a Space in a Catalog to deprecate a published Product.

- **[Retiring a Product](#)**

You can retire a published or deprecated Product by using the Manage Products page within a Catalog in API Manager. When a Product is retired, all associated APIs are taken offline, and any subscriptions are deleted. The syndication feature in IBM API Connect means that you can also use the Manage Products page within a Space in a Catalog to retire a Product.

- **[Re-staging a Product](#)**

If a Product in a Catalog has been retired, you can re-stage the Product so that it can re-enter the Product lifecycle. A re-staged Product moves to the Staged state, from where it can be published. The syndication feature in IBM API Connect means that you can also re-stage a Product within a Space in a Catalog.

- **[Working with the subscriptions in a Product](#)**

You can work with the subscriptions to the Plans in a specific Product. Plan subscriptions are created by application developers in the Developer Portal.

- **[Updating the gateway services for a Product](#)**

When you stage or publish a draft Product to a Catalog, you specify the gateway services at which the APIs in the Product are to be made available. However, you can later update the selected gateway services after a Product has been stage or published.

- **[Removing a Product from a Catalog](#)**

You can remove a Product from a specific Catalog by using the Products view within a Catalog in API Manager. The syndication feature in IBM API Connect means that you can also use the Products view within a Space to remove a Product from a Catalog.

- **[Approving Product lifecycle and subscription requests](#)**

To approve or decline requests to change the lifecycle state of a Product, and requests by application developers to subscribe to a Plan, use the Tasks page in the Catalog that contains the associated Product in API Manager. The syndication feature in IBM API Connect means that you can also use the Tasks page within a Space in a Catalog to approve or decline requests.

---

## Considerations when changing a Product lifecycle with billing integration

There are special considerations for lifecycle operations for Products that contain a billing integration in IBM® API Connect.

For example, if you release new versions of your Product that contain pricing changes, you will need to define the upgrade pathways for your existing subscribers. The following scenarios detail the special considerations for Product lifecycle changes:

### Deprecating

Preparing an existing Product to be removed from production. You can define a replacement Product by using the Set Migration Target option in the Catalog. API consumers will then see a Migrate this subscription message in the Developer Portal that they can click to upgrade their subscription to the migration target. If the upgrade target is a paid Plan, they must enter a payment method before upgrading. Upgrades by API consumers from a free Plan to a paid Plan are supported. For more information about deprecating Products, see [Deprecating a Product](#).

### Migrating

Moving subscriptions from one Product to a different Product, and from a free Plan to a paid Plan, is not supported. Instead, see the superseding scenario.

### Replacing

Replacing a Product with a newer version of the same Product, and automatically upgrading all existing subscribers to the new Product. You must map which Plans from your new Product correspond to which Plans in your old Product. You cannot map a free Plan to a paid Plan in the replace operation. For more information

about replacing Products, see [Replacing a Product with another Product](#).

#### Retiring

Removing a Product or Plan from production, and canceling all of the existing subscriptions and their billing cycles. For more information about retiring Products, see [Retiring a Product](#).

#### Superseding

Deprecating the old Product, and at the same time defining its migration target to a newer version of itself. Retaining the existing Product and Plan for subscribers who would like to keep it, but offering a different Product and Plan for the updated version of a Product. API consumers will then see a Migrate this subscription message in the Developer Portal that they can click to upgrade their subscription to the migration target. If the upgrade target is a paid Plan, they must enter a payment method before upgrading. Upgrades by API consumers from a free Plan to a paid Plan are supported. For more information about superseding Products, see [Superseding a Product with another Product](#).

Note: You can use the Set Migration Target option in the Catalog on a Product that isn't being deprecated. For example, if you wanted to give customers the option to move to a different Product without affecting the original Product.

While these follow the same basic procedures that are described in [Managing your Products](#) for Products and Plans without billing, there are some considerations that only apply to Products that contain paid Plans.

## You cannot migrate a customer directly from a free Plan to a Plan with billing

---

Customers who subscribe to Plans with billing must set up their billing information in the Developer Portal. If they subscribe only to free Plans, they don't have billing accounts.

There are two ways to migrate customers from a free Plan to a paid Plan:

- Supersede the free Product Plan and provide a migration target to a paid Product Plan. This deprecates the free Product Plan so no new users can subscribe to it, and gives the customer a Migrate this subscription message in the Developer Portal to move to the new Product Plan.
- Set a migration target to a paid Product Plan. In this case the free Product Plan is not deprecated, but it still gives the customer the Migrate this subscription message in the Developer Portal for moving to the new Product Plan. With this method, new customers can still subscribe to the free Product Plan.

Both of these methods require action from the customer before they move from the free Product Plan to the paid Product Plan.

Note: Free trial days do not apply to customers when they migrate. They are available only for new customers.

## You must map the existing Plans to the new Plans when you supersede or replace a Product

---

When you replace or supersede a Plan with billing with another Plan, you must map the Plans from the Product that is being removed to the Plans that are in the new Product. It is convenient if there are the same number of Plans for each, but you can map more than one Plan from the existing Product to one Plan in the new Product.

## Billing charges are prorated when subscriptions are migrated

---

When customers change their subscriptions to a Plan that has a different price, their next monthly invoice will contain prorated charges, depending on how long they spent on different billing Plans. For more information about how Stripe handles proration, see <https://stripe.com/docs/billing/subscriptions/prorations>.

## Related reference

---

- [Troubleshooting your billing configuration](#)

## Publishing a Product

---

APIs become accessible when a Product is published and made visible on the IBM® API Connect Developer Portal for use by application developers. A Product can be published to selected communities of application developer organizations, and the Plans within the Product can be used to tailor access and visibility further.

## Before you begin

---

You must stage a Product before it can be published. For more information about staging Products, see [Staging a Product](#).

Note: If you want to publish a LoopBack project, you must publish both the APIs (by publishing the Products that contain the APIs) and the associated applications so the project can be run. For more information about publishing LoopBack applications, see [Publishing APIs and applications](#).

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Catalog that contains the Product. If you have View permission for Products, you have read-only access to the Product management page. For information on configuring Product management permissions for a Catalog, see [Creating and configuring Catalogs](#).

The syndication feature in IBM API Connect means that Products can be contained within a Space in a Catalog. In this case, to complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Space that contains the Product. For information on configuring Product management permissions for a Space, see [Managing user access in a Space](#).

## About this task

---

A community is a collection of consumer organizations, used to control which organizations have access to Products and Plans without having to assign access on an individual basis. A Product can be published to selected communities, which means that only application developers within those organizations contained within the community can see the Product on the Developer Portal and obtain application keys to access it. A Product can alternatively be published to all communities. Communities are used to restrict the visibility and accessibility of APIs, for example to particular business partners, internal organizations, or other groups of application developers.

Note: If you want to update a Product, you must republish it. The `products:update` command only updates a Product's metadata.

## Procedure

---

You can publish a product in any of the following ways:

- Publish a new Product.
  - Replace a Product with another Product.
  - Supersede a Product with another Product
- For details of the ways in which you can publish a Product, see the following subtopics:

- **[Publishing a new Product](#)**  
Publish a Product from within its containing Catalog in API Manager. The syndication feature in IBM API Connect means that you can also publish a Product from within its containing Space in a Catalog.
- **[Replacing a Product with another Product](#)**  
You can replace a published Product in IBM API Connect with another Product, and automatically migrate subscribers to the new Product, by using the API Manager.
- **[Superseding a Product with another Product](#)**  
You can supersede a published Product with another Product by using the API Manager.

---

## Publishing a new Product

Publish a Product from within its containing Catalog in API Manager. The syndication feature in IBM® API Connect means that you can also publish a Product from within its containing Space in a Catalog.

---

### Before you begin

The Product that you are publishing must be in the Staged or Deprecated state.

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Catalog that contains the Product. If you have View permission for Products, you have read-only access to the Product management page. For information on configuring Product management permissions for a Catalog, see [Creating and configuring Catalogs](#).

The syndication feature in IBM API Connect means that Products can be contained within a Space in a Catalog. In this case, to complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Space that contains the Product. For information on configuring Product management permissions for a Space, see [Managing user access in a Space](#).

---

### About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

If you publish a Product to a non-development Catalog, the Product that is published is an independent and fixed copy of the version of the Product that you chose to stage. Editing the Product through the Products page will not affect the published Product. For this reason, it is recommended that when you stage a Product, you then create a new version of the Product to edit in future, so as to avoid confusion regarding the properties of the published Product. For more information on creating new versions of your Product, see [Creating a new version of your Product](#).

An exception is that if you publish a Product to a development Catalog, editing it through the Products page will enable you to re-stage and publish the same version of the Product. For information on how to create a development Catalog, see [Creating and configuring Catalogs](#).



Although you can publish to a development Catalog, the development Catalog should be used only for testing purposes. Similarly, a Developer Portal created from a development Catalog must be used for testing purposes only, and not for production use. For more information on Catalogs, see [Working with Catalogs](#).

Note: All references in this topic to a Catalog can also be applied to a Space in a Catalog, unless specified otherwise. For more information about Spaces, see [Using syndication in API Connect](#).

---


### Procedure

To publish a Product, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.  
The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.
2. If the Product that you want to work with is contained within a Space, select the required Space by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.The Products page of the Space opens, and all of the Products available in that Space are displayed.
3. Alongside the Product version that you want to work with, click the options icon  and then click Publish.  
The Confirm Visibility Settings page opens.
4. Specify the following options:
  - The users that the Product is visible to  
You can choose Public users, Authenticated users, or Custom.
    - Custom - You can use the Type to add... field to search for organizations or communities that you want your Product to be visible to.
  - Who can subscribe to the Product  
You can add or remove one or more consumer organizations or communities.
    - Custom - You can use the Type to add... field to search for organizations or communities that you want to allow to subscribe to your Product.
5. Click Publish, then click Confirm to proceed with the publish operation.  
If approval is required to publish Products in this Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is published when the request is approved. If approval is not required, the Product version is published immediately, and moves to the Published state. For information on configuring

Product lifecycle approvals for a Catalog, see [Creating and configuring Catalogs](#). For information on approving requests, see [Approving Product lifecycle and subscription requests](#).

Note:

- Approval for Product lifecycle state changes in a Catalog is disabled by default. You must explicitly enable the Product lifecycle state changes that you want to enforce.
- Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
- You can view the history of Product lifecycle requests and approvals, by clicking the options icon  alongside the Product that you want to work with, and selecting View Approval History.

## Results

---

Your Product is in the Published state.

Your Product is published to your Catalog and available to your specified organizations or communities. Application developers within the groups you selected can see and use the APIs within the Product.

Any application developer requests to use your Product are displayed on the Approvals tab in the containing Catalog, where you can decline or accept the request.

## Related tasks

---

- [Deprecating a Product](#)

## Related information

---

- [Staging a Product](#)

# Replacing a Product with another Product

---

You can replace a published Product in IBM® API Connect with another Product, and automatically migrate subscribers to the new Product, by using the API Manager.

## Before you begin

---

The Product to be replaced must be in the Published state, and the replacement Product must be in the Staged, Published, or Deprecated state.

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Catalog that contains the Product. If you have View permission for Products, you have read-only access to the Product management page. For information on configuring Product management permissions for a Catalog, see [Creating and configuring Catalogs](#).

The syndication feature in IBM API Connect means that Products can be contained within a Space in a Catalog. In this case, to complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Space that contains the Product. For information on configuring Product management permissions for a Space, see [Managing user access in a Space](#).

## About this task

---

When you replace a Product with another Product, the following actions are taken:

- The replacement Product is published.
- If the visibility and subscribability settings in the replacement Product are such that access is the same as, or less restrictive than, the original Product, the settings in the replacement Product are used. If the settings in the replacement Product are more restrictive, meaning that fewer consumer organizations can see or subscribe to the Product, the replace operation fails. For more information on visibility and subscribability settings, see [Changing the availability of a Product](#).
- The subscribers to the original Product are migrated to the replacement Product.  
Note: Customers cannot be migrated automatically from a free Plan to a paid Plan. To move your customers from a free Plan to a paid Plan, you can supersede the product with a new product and set a migration target to the paid Plan. The customers then select a button to migrate and must enter their credit card information before the process is complete. For more information, see [Considerations when changing a Product lifecycle with billing integration](#).
- The original Product is moved to the Retired state. Products in the Retired state are removed from the Developer Portal; they are no longer visible to the application developers, and any subscriptions to them are canceled. However, the Product can be staged again later if required.



Although you can publish to a development Catalog, the development Catalog should be used only for testing purposes. Similarly, a Developer Portal created from a development Catalog must be used for testing purposes only, and not for production use. For more information on Catalogs, see [Working with Catalogs](#).

Note: All references in this topic to a Catalog, can also be applied to a Space in a Catalog, unless specified otherwise.

## Procedure

---


To replace a Product, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.  
The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.
2. If the Product that you want to work with is contained within a Space, select the required Space by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.The Products page of the Space opens, and all of the Products available in that Space are displayed.
3. Alongside the Product version that you want to replace, click the options icon , then click Replace.  
The Replace window opens.

4. Select the replacement Product, then click Next.
5. From the drop-down list, select which Plans from your new Product correspond to Plans in your old Product.
6. Click Replace.

If approval is required to replace Products in this Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is replaced when the request is approved. If approval is not required, the Product is replaced immediately. For information on configuring Product lifecycle approvals for a Catalog, see [Creating and configuring Catalogs](#). For information on approving requests, see [Approving Product lifecycle and subscription requests](#).

Note:

- Approval for Product lifecycle state changes in a Catalog is disabled by default. You must explicitly enable the Product lifecycle state changes that you want to enforce.
- Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
- You can view the history of Product lifecycle requests and approvals, by clicking the options icon  alongside the Product that you want to work with, and selecting View Approval History.

## Results

---

Your new Product is in the Published state, and the original Product is in the Retired state.

Your new Product is published to your preferred organizations or communities. Application developers within the groups you selected can see and use the APIs within the Product.

Any application developer requests to use your Product are displayed on the Approvals tab in the containing Catalog, where you can decline or accept the request.

## Related concepts

---

- [The Product lifecycle](#)

## Related information

---

- [Staging a Product](#)

# Superseding a Product with another Product

---

You can supersede a published Product with another Product by using the API Manager.

## Before you begin

---

The Product to be superseded must be in the Published state. The superseding Product must be in the Staged, Published, or Deprecated state.

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Catalog that contains the Product. If you have View permission for Products, you have read-only access to the Product management page. For information on configuring Product management permissions for a Catalog, see [Creating and configuring Catalogs](#).

The syndication feature in IBM® API Connect means that Products can be contained within a Space in a Catalog. In this case, to complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Space that contains the Product. For information on configuring Product management permissions for a Space, see [Managing user access in a Space](#).

## About this task

---

When you supersede a Product with another Product, the following actions are taken:

- The superseding Product is published.
- If the visibility and subscribability settings in the superseding Product are such that access is the same as, or less restrictive than, the original Product, the settings in the superseding Product are used. If the settings in the superseding Product are such that access is more restrictive, meaning that fewer consumer organizations can see or subscribe to the Product, the supersede operation fails. For more information on visibility and subscribability settings, see [Changing the availability of a Product](#).
- The original Product is moved to the Deprecated state.
- The application developers that are already subscribed to the now deprecated Product can continue to use it, but no new developers can subscribe to the Product. In the Developer Portal the application developers will see a Migrate this subscription message, which they can click to upgrade their subscription to the migration target.
- The deprecated product can be published again if required.


Although you can publish to a development Catalog, the development Catalog should be used only for testing purposes. Similarly, a Developer Portal created from a development Catalog must be used for testing purposes only, and not for production use. For more information on Catalogs, see [Working with Catalogs](#).

Note: All references in this topic to a Catalog, can also be applied to a Space in a Catalog, unless specified otherwise.

## Procedure


---

To supersede a Product, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.  
The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.
2. If the Product that you want to work with is contained within a Space, select the required Space by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.




The Products page of the Space opens, and all of the Products available in that Space are displayed.

3. Alongside the Product version that you want to supersede, click the options icon  and then click Supersede.
4. Select the superseding Product and then click Next.
5. From the drop-down list, select which Plans from your new Product correspond to Plans in your old Product.
6. Click Supersede.

If approval is required to supersede Products in this Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is superseded when the request is approved. If approval is not required, the Product is superseded immediately. For information on configuring Product lifecycle approvals for a Catalog, see [Creating and configuring Catalogs](#). For information on approving requests, see [Approving Product lifecycle and subscription requests](#).

Note:

- Approval for Product lifecycle state changes in a Catalog is disabled by default. You must explicitly enable the Product lifecycle state changes that you want to enforce.
- Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
- You can view the history of Product lifecycle requests and approvals, by clicking the options icon  alongside the Product that you want to work with, and selecting View Approval History.

## Results

---

Your superseding Product is in the Published state.

The Product that was superseded is in the Deprecated state.

Your superseding Product is published to your preferred organizations or communities. Application developers within the groups you selected can see and use the APIs within the Product. The original Product is deprecated.

Any application developer requests to use your Product are displayed on the Approvals tab in the containing Catalog, where you can decline or accept the request.

Existing customers remain subscribed to the deprecated product, and any billing cycles for subscriptions to paid plans continue uninterrupted. For more information, see [Considerations when changing a Product lifecycle with billing integration](#).

## Related concepts

---

- [The Product lifecycle](#)

## Related tasks

---

- [Deprecating a Product](#)
- [Publishing a new Product](#)

# Migrating application subscriptions to another Product

By using the Products view within a Catalog in API Manager, you can migrate application subscriptions between Products. The syndication feature in IBM® API Connect means that you can also use the Products view within a Space in a Catalog to migrate application subscriptions.

## Before you begin

---

Your Product that you are migrating subscriptions from must be in one of the following states:

- Published
- Deprecated

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Catalog that contains the Product. If you have View permission for Products in this Catalog, you will have read-only access to the Product management page. For information on configuring Product management permissions for a Catalog, see [Creating and configuring Catalogs](#).

The syndication feature in IBM API Connect means that Products can be contained within a Space in a Catalog. In this case, to complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Space that contains the Product. For information on configuring Product management permissions for a Space, see [Managing user access in a Space](#).

## About this task


---


Application developers initially subscribe their applications to one or more Plans in Products by using the Developer Portal. For more information, see [Exploring APIs and Products in the Developer Portal](#). However, by using API Manager, for a chosen Product you can migrate one or more of its application subscriptions from one Plan to a Plan in another Product.

## Procedure

---

To migrate Product subscriptions, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.  
The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.
2. If the Product that you want to work with is contained within a Space, select the required Space by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.  
The Products page of the Space opens, and all of the Products available in that Space are displayed.

3. Alongside the Product version that you want to work with, click the options icon  and then click Migrate Subscriptions.
4. Select the Plan from which you want to migrate subscriptions, then click Next.
5. Select the subscriptions that you want to migrate, then click Next.
6. Select the Product that you want to migrate the subscriptions to, then click Next.
7. Select the Plan that you want to migrate the application subscriptions to, and then click Migrate.

---

## Results

The target Product has the migrated subscriptions, and those subscriptions are removed from the Product from which they were migrated.

---

## Related tasks

- [Working with Spaces](#)


---

# Migrating application subscribers to new Product versions

When new versions of Products are created in IBM® API Connect, there are a number of ways that subscribers can be moved to the Plans of the new Product.

---

## About this task

You can move users to the new Plans in different ways, depending on your Product strategy, by using the options icon  alongside the Product in the associated Catalog in the API Manager UI.

---

## Procedure

To automatically migrate all existing subscribers to a new Product, for example, when you are applying fixes to a Product:

- You should Replace the original version of the Product with a new version of the Product.
  - The replacement Product is Published.
  - The original Product is Retired.
  - The subscribers to the original Product are automatically migrated to the replacement Product.

Note: Customers cannot be migrated automatically from a free Plan to a paid Plan. To move your customers from a free Plan to a paid Plan, you can supersede the product with a new product and set a migration target to the paid Plan. The customers then select a button to migrate and must enter their credit card information before the process is complete. For more information, see [Considerations when changing a Product lifecycle with billing integration](#).

For more information about replacing Products, see [Replacing a Product with another Product](#).

To encourage subscribers to move to a new Product, and stop new users from subscribing to the original Product, for example, if an enhancement or new feature is added:

- You should Supersede the original version of the Product with a new version of the Product.
  - The superseding Product is Published.
  - The original Product is Deprecated.
  - The application developers that are already subscribed to the now deprecated Product can continue to use it, but no new developers can subscribe to the Product. In the Developer Portal the subscribers will see a Migrate this subscription message, which they can click to upgrade their subscription to the migration target.
  - If the migration target is a paid Plan, subscribers must enter a payment method before they can upgrade. Upgrades by subscribers from a free Plan to a paid Plan are supported.

For more information about superseding Products, see [Superseding a Product with another Product](#).

To prepare a Product to be removed from production, but leave the existing subscriptions as they are:

- You should Deprecate the original Product.
  - The Product is Deprecated.
  - The application developers that are already subscribed to the now deprecated Product can continue to use it, but no new developers can subscribe to the Product.
  - You can define a replacement Product by using the Set Migration Target option in the Catalog. Application developers will then see a Migrate this subscription message in the Developer Portal that they can click to upgrade their subscription to the migration target. If the upgrade target is a paid Plan, they must enter a payment method before upgrading. Upgrades by API consumers from a free Plan to a paid Plan are supported.

For more information about deprecating Products, see [Deprecating a Product](#).

To give subscribers the option to move to a different Product, but without affecting the original Product:

- You can use the Set Migration Target option in the Catalog on a Product that isn't being deprecated or superseded. For more information, see [Managing your Products](#).

---

# Changing the availability of a Product

You can change the availability of a Product and the associated Plans by using the Products view within a Catalog in API Manager. The syndication feature in IBM® API Connect means that you can also use the Products view within a Space in a Catalog to change the availability of a Product.

---

## Before you begin

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Catalog that contains the Product. If you have View permission for Products, you have read-only access to the Product management page. For information on configuring Product management permissions for a Catalog, see [Creating and configuring Catalogs](#).



The syndication feature in IBM API Connect means that Products can be contained within a Space in a Catalog. In this case, to complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Space that contains the Product. For information on configuring Product management permissions for a Space, see [Managing user access in a Space](#).

Note: All references in this topic to a Catalog can also be applied to a Space in a Catalog, unless specified otherwise. For more information about Spaces, see [Using syndication in API Connect](#).

## Procedure

---

To change the availability of a Product, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.  
The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.
2. If the Product that you want to work with is contained within a Space, select the required Space by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.The Products page of the Space opens, and all of the Products available in that Space are displayed.
3. Alongside the Product version that you want to work with, click the options icon , then click Edit visibility.  
The Visibility page opens.
4. In the Visibility section, specify the users that you want the Product to be visible to. You can choose Public to make the Product visible to all users, Authenticated to make the Product available to users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that you want the Product to be visible to.  
If you select Custom, use the Type to add organizations field to search for the consumer organizations and consumer organization groups that you want the Product to be visible to. For information about how to create and manage consumer organizations and consumer organization groups, see [Working with consumer organizations](#).  
Use the Temporarily disable visibility check box to toggle whether the Product is or is not visible, without having to change other visibility settings.
5. In the Subscribability section, specify the users that can subscribe to the Plans in the Product. You can choose Authenticated to make the Plans in the Product subscribable by users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that can subscribe to the Plans in the Product.  
If you select Custom, use the Type to add organizations field to search for the consumer organizations and consumer organization groups that can subscribe to the Plans in the Product. If you selected custom visibility, only the consumer organizations and consumer organization groups selected there are available for adding to the custom subscribability list. For information about how to create and manage consumer organizations and consumer organization groups, see [Working with consumer organizations](#).  
Use the Temporarily disable subscribability check box to toggle whether the Product can or cannot be subscribed to, without having to change other subscription settings.
6. Click Save when done.  
The Product version is configured with the modified availability settings.  
Note:
  - Changing who can view or subscribe to a Product does not affect the settings of the draft Product. The change applies only in the Catalog that was selected in Step 1.
  - Changing who can view or subscribe to a Product does not affect existing subscriptions.

## Results

---

Your Product remains in the same lifecycle state, now with new availability settings.

## Related tasks

---

- [Working with Spaces](#)
- [Configuring Product visibility in a Catalog or a Space](#)

## Deprecating a Product

---

You can deprecate a published Product by using the Manage Products page within a Catalog in API Manager. When you deprecate a Product, application developers that are already subscribed to the Product can continue to use it, but no new developers can subscribe to the Product. The syndication feature in IBM® API Connect means that you can also use the Manage Products page within a Space in a Catalog to deprecate a published Product.

## Before you begin

---

The Product that you are deprecating must be in the Published state.




To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Catalog that contains the Product. If you have View permission for Products, you have read-only access to the Product management page. For information on configuring Product management permissions for a Catalog, see [Creating and configuring Catalogs](#).

The syndication feature in IBM API Connect means that Products can be contained within a Space in a Catalog. In this case, to complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Space that contains the Product. For information on configuring Product management permissions for a Space, see [Managing user access in a Space](#).

## Procedure

---

To deprecate a Product, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.  
The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.
  2. If the Product that you want to work with is contained within a Space, select the required Space by completing the following steps:
    - a. Click the Spaces tab.
    - b. Select the Space that you want to work with.The Products page of the Space opens, and all of the Products available in that Space are displayed.
  3. Alongside the Product version that you want to work with, click the options icon , then click Deprecate.  
The Deprecate Product window opens.
  4. Click Confirm to deprecate the Product.  
If approval is required to deprecate Products in this Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is deprecated when the request is approved. If approval is not required, the Product is deprecated immediately. For information on configuring Product lifecycle approvals for a Catalog, see [Creating and configuring Catalogs](#). For information on approving requests, see [Approving Product lifecycle and subscription requests](#).
- Note:
- Approval for Product lifecycle state changes in a Catalog is disabled by default. You must explicitly enable the Product lifecycle state changes that you want to enforce.
  - Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
  - You can view the history of Product lifecycle requests and approvals, by clicking the options icon  alongside the Product that you want to work with, and selecting View Approval History.

## Results

---

Your Product is in the Deprecated state.

## Related concepts

---

- [Using syndication in API Connect](#)

## Related tasks

---

- [Working with Spaces](#)

## Retiring a Product

You can retire a published or deprecated Product by using the Manage Products page within a Catalog in API Manager. When a Product is retired, all associated APIs are taken offline, and any subscriptions are deleted. The syndication feature in IBM® API Connect means that you can also use the Manage Products page within a Space in a Catalog to retire a Product.

## Before you begin

---

The Product that you are retiring must be in the Published or Deprecated state.



To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Catalog that contains the Product. If you have View permission for Products, you have read-only access to the Product management page. For information on configuring Product management permissions for a Catalog, see [Creating and configuring Catalogs](#).


The syndication feature in IBM API Connect means that Products can be contained within a Space in a Catalog. In this case, to complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Space that contains the Product. For information on configuring Product management permissions for a Space, see [Managing user access in a Space](#).

## Procedure

---

To retire a Product, complete the following steps.

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.  
The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.
  2. If the Product that you want to work with is contained within a Space, select the required Space by completing the following steps:
    - a. Click the Spaces tab.
    - b. Select the Space that you want to work with.The Products page of the Space opens, and all of the Products available in that Space are displayed.
  3. Alongside the Product version that you want to work with, click the options icon  and then click Retire. The Retire Product window opens.
  4. Click Confirm.  
If approval is required to retire Products in this Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is retired when the request is approved. If approval is not required, the Product is retired immediately. For information on configuring Product lifecycle approvals for a Catalog, see [Creating and configuring Catalogs](#). For information on approving requests, see [Approving Product lifecycle and subscription requests](#).
- Note:
- Approval for Product lifecycle state changes in a Catalog is disabled by default. You must explicitly enable the Product lifecycle state changes that you want to enforce.

- Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
- You can view the history of Product lifecycle requests and approvals, by clicking the options icon  alongside the Product that you want to work with, and selecting View Approval History.

## Results

---

Your Product is in the Retired state. It is also removed from the Developer Portal, is no longer visible to the application developers, and any subscriptions to it are deleted. You can stage the Product again later if required; for details, see [Staging a Product](#). If the Product has any monetized Plans, the billing cycles for any subscriptions are canceled as well. For more information, see [Considerations when changing a Product lifecycle with billing integration](#).

## Related concepts

---

- [Using syndication in API Connect](#)

## Related tasks

---

- [Working with Spaces](#)

## Re-staging a Product

---

If a Product in a Catalog has been retired, you can re-stage the Product so that it can re-enter the Product lifecycle. A re-staged Product moves to the Staged state, from where it can be published. The syndication feature in IBM® API Connect means that you can also re-stage a Product within a Space in a Catalog.

## Before you begin

---

The Product that you are re-staging must be in the Retired state.

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Catalog that contains the Product. If you have View permission for Products, you have read-only access to the Product management page. For information on configuring Product management permissions for a Catalog, see [Creating and configuring Catalogs](#).

The syndication feature in IBM API Connect means that Products can be contained within a Space in a Catalog. In this case, to complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Space that contains the Product. For information on configuring Product management permissions for a Space, see [Managing user access in a Space](#).

## About this task



---

You stage a Product so that the appropriate approvals, internally within the organization, can be given for it to then be published. For information on configuring Product lifecycle approvals for a Catalog, see [Creating and configuring Catalogs](#). For information on approving requests, see [Approving Product lifecycle and subscription requests](#). For information on publishing a Product, see [Publishing a Product](#).

## Procedure


---

To re-stage a Product, complete the following steps.

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with. The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.
2. If the Product that you want to work with is contained within a Space, select the required Space by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.The Products page of the Space opens, and all of the Products available in that Space are displayed.
3. Alongside the Product version that you want to work with, click the options icon  and then click Re-stage. The Re-stage Product window opens.
4. Click Confirm.

If approval is required to stage Products in this Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is re-staged when the request is approved. If approval is not required, the Product is re-staged immediately. For information on configuring Product lifecycle approvals for a Catalog, see [Creating and configuring Catalogs](#). For information on approving requests, see [Approving Product lifecycle and subscription requests](#).

Note:

  - Approval for Product lifecycle state changes in a Catalog is disabled by default. You must explicitly enable the Product lifecycle state changes that you want to enforce.
  - Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
  - You can view the history of Product lifecycle requests and approvals, by clicking the options icon  alongside the Product that you want to work with, and selecting View Approval History.

## Results

---

Your Product is in the Staged state, from where it can be published; see [Publishing a new Product](#).

## Related concepts

---

- [Using syndication in API Connect](#)

## Related tasks

---

- [Working with Spaces](#)









## Working with the subscriptions in a Product

You can work with the subscriptions to the Plans in a specific Product. Plan subscriptions are created by application developers in the Developer Portal.

### Procedure

---

To work with the Plan subscriptions in a Product, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with. The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.
2. Optional: If Spaces are enabled in the Catalog, select the Space that you want to work with by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.For more information about enabling Spaces, see [Using syndication in API Connect](#).
3. Alongside the Product version that you want to work with, click the options icon  and then click View Subscriptions. A window opens listing all subscriptions to the Plans in the selected Product.
4. Alongside the subscription that you want to work with, click the options icon . You can then select from the following options:
  - View Consumer Organization: a window opens listing the consumer organization that the subscribing application belongs to. If required, you can click the options icon  to work with the consumer organization. For details of the available options, see [Working with consumer organizations](#).
  - View Application: a window opens listing the subscribing application. If required, you can click the options icon  to work with the application. For details of the available options, see [Working with developer applications](#).
  - View Plan: a window opens listing the Plan that the application is subscribed to. If required, you can click the options icon  to work with the Plan. For details of the available options, see [Working with Plans](#).
  - View Product: a window opens listing the Product that contains the Plan that the application is subscribed to. If required, you can click the options icon  to work with the Product. For details of the available options, see [Managing your Products](#).
5. To display the Client ID's for a particular Product subscription, click the information icon  in the Application column for the subscription that you require.

## Updating the gateway services for a Product

When you stage or publish a draft Product to a Catalog, you specify the gateway services at which the APIs in the Product are to be made available. However, you can later update the selected gateway services after a Product has been stage or published.

### Before you begin

---

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Catalog that contains the Product. If you have View permission for Products, you have read-only access to the Product management page. For information on configuring Product management permissions for a Catalog, see [Creating and configuring Catalogs](#).

The syndication feature in IBM® API Connect means that Products can be contained within a Space in a Catalog. In this case, to complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Space that contains the Product. For information on configuring Product management permissions for a Space, see [Managing user access in a Space](#).

### About this task

---

When you stage or publish a draft Product to a Catalog, you can either choose to make the APIs in the Product available at all the gateway services that match the gateway service type of the Product, either DataPower® API Gateway or DataPower Gateway (v5 compatible), or can you select specific gateway services of that type. The APIs in the Product can then be called at those gateway service endpoints.

However, after a Product has been staged or published you can later update the gateway services to change the selections.

For more information on staging and publishing a Product, see [Staging a draft Product](#) and [Publishing a draft Product](#).


For more information on the gateway service types, see [API Connect gateway types](#).

### Procedure

---

To update the gateway services for a Product, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with. The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.

2. If the Product that you want to work with is contained within a Space, select the required Space by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.The Products page of the Space opens, and all of the Products available in that Space are displayed.
3. Alongside the Product version that you want to work with, click the options icon , then click Update gateway services. The Select Gateway Services window opens.
4. Select the required gateway services, then click Save.

---

## Removing a Product from a Catalog

You can remove a Product from a specific Catalog by using the Products view within a Catalog in API Manager. The syndication feature in IBM® API Connect means that you can also use the Products view within a Space to remove a Product from a Catalog.

### Before you begin



The Product that you are removing from the Catalog must be in the Staged or Retired state. For more information, see [Staging a Product](#) or [Retiring a Product](#).

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Catalog that contains the Product. If you have View permission for Products, you have read-only access to the Product management page. For information on configuring Product management permissions for a Catalog, see [Creating and configuring Catalogs](#).

The syndication feature in IBM API Connect means that Products can be contained within a Space in a Catalog. In this case, to complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Space that contains the Product. For information on configuring Product management permissions for a Space, see [Managing user access in a Space](#).

### Procedure

To remove a Product from a Catalog, complete the following steps.

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with. The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.
2. If the Product that you want to work with is contained within a Space, select the required Space by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.The Products page of the Space opens, and all of the Products available in that Space are displayed.
3. Alongside the Product version that you want to work with, click the options icon , then click Delete. The Delete Product window opens.
4. Click Confirm to delete the Product.

### Results

Your Product is removed from the Catalog. You can stage the Product again if required; for details, see [Staging a Product](#). If your Product was contained within a Space in a Catalog, your Product is removed from both the Space and the Catalog.

### Related concepts

- [Using syndication in API Connect](#)

### Related tasks

- [Working with Spaces](#)

---

## Approving Product lifecycle and subscription requests

To approve or decline requests to change the lifecycle state of a Product, and requests by application developers to subscribe to a Plan, use the Tasks page in the Catalog that contains the associated Product in API Manager. The syndication feature in IBM® API Connect means that you can also use the Tasks page within a Space in a Catalog to approve or decline requests.

### Before you begin


To see lifecycle change requests for a Product, you must either be the owner of the API provider organization, or you must be assigned a user role that has permission to view or manage Products in the Catalog that contains the Product. For information on configuring Product management permissions for a Catalog, see [Creating and configuring Catalogs](#).

The syndication feature in API Connect means that Products can be contained within a Space in a Catalog. In this case, to see lifecycle change requests for a Product, you must either be the owner of the API provider organization, or you must be assigned a user role that has permission to view or manage Products in the Space that contains the Product. For information on configuring Product management permissions for a Space, see [Managing user access in a Space](#).

### About this task

If approvals for Product lifecycle changes are enabled for a Catalog, then an attempt to change the lifecycle state of a Product results in an approval request being sent. This request is displayed in the Tasks page in the Catalog, from where the request can be approved or declined. The authority to approve Product lifecycle state changes is restricted to users in specified roles. For information on configuring Product lifecycle approvals for a Catalog, see [Creating and configuring Catalogs](#).

Note:


- Approval for Product lifecycle state changes in a Catalog is disabled by default. You must explicitly enable the Product lifecycle state changes that you want to enforce.
- Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
- You can view the history of Product lifecycle requests and approvals, by clicking the options icon  alongside the Product that you want to work with, and selecting View Approval History.

If a Product is set to require approval for subscription by application developers, then an attempt by an application developer to subscribe to the Product results in an approval request being sent. This request is displayed in the Approvals tab in the Catalog that contains the associated Product in API Manager, from where the request can be approved or declined.


## Procedure

---

To work with a Product lifecycle change request, complete the following steps.

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.  
The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.
2. If the Product that you want to work with is contained within a Space, select the required Space by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.The Products page of the Space opens, and all of the Products available in that Space are displayed.
3. Click the Tasks tab, then locate the Product lifecycle change request that you want to deal with.
4. Click the Approve or Decline as required.  
Note: You can approve or decline lifecycle change requests only if you have a user role that has permission to approve Product publish requests in the Catalog in which the changes will take place.  
For information on configuring Product management permissions for a Catalog, see [Creating and configuring Catalogs](#).

To work with a subscription request, complete the following steps.

5. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.  
The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.
6. If the Product that you want to work with is contained within a Space, select the required Space by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.The Products page of the Space opens, and all of the Products available in that Space are displayed.
7. Click the Tasks tab, then locate the subscription requests that you want to deal with.  
Note: To see subscription approvals, you must either be the owner of the API provider organization, or you must be assigned a user role that has permission to view or approve subscription requests. For information on creating and assigning user roles, see [Administering user access](#).
8. Click the Approve or Decline as required.  
Note: The options to approve or decline a subscription request are available only if you are assigned a user role that has permission to approve subscription requests. For information on configuring permissions for a Catalog, see [Creating and configuring Catalogs](#) For information on assigning user roles, see [Managing Catalog membership](#).

## Results

---

For a subscription request, an email is sent confirming whether you approved or declined the request. For a request to change the lifecycle state of a Product, no email is sent.

## Related concepts

---

- [Using syndication in API Connect](#)

## Related tasks

---

- [Working with Spaces](#)


## Working with APIs

For a specific Product, you can work with the APIs that are contained in the Plans in that Product.





## Procedure

---

To work with the APIs in a Product, complete the following steps:

1. List the Plans in the Product:
  - a. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
  - b. If Spaces are enabled in the Catalog, select the Space that you want to work with by completing the following steps:
    - i. Click the Spaces tab.
    - ii. Select the Space that you want to work with.



- c. If the Products page is not already displayed, click the Products tab.
  - d. Alongside the Product version that you want to work with, click the options icon , then click View Plans.  
A window opens listing the Plans in the Product.
2. Alongside the required Plan, click View APIs.  
A window opens listing the APIs in the Plan.
  3. Alongside the required API, click the options icon .  
You can then select from the following options:
    - Take Offline: if you take an API offline, the API is no longer available to be called. When required, you can make the API available again by selecting Online.
    - View Products: a window opens listing all the Products that contain the API. If required, you can click the options icon  alongside a Product to work with that Product. For details of the available options, see [Managing your Products](#).
    - View Plans: a window opens listing all Plans, across all Products, that contain the API. If required, you can click the options icon  alongside a Plan to work with that Plan. For details of the available options, see [Working with Plans](#).

---

## Working with Plans

After a Product has been published to a Catalog, you can work with the Plans in that Product.

---

### Before you begin

For details on configuring a Plan in a Product, see [Editing a draft Product](#).







For details on Publishing a Product to a Catalog, see [Publishing a Product](#).

To complete the tasks that are described in this topic, you must either be the owner of the API provider organization, or you must be assigned a role that has either the Product\_>Manage or the Product\_>View permission for the Catalog. For more information, see [Adding provider organization users and assigning roles](#) and [Managing Catalog membership](#). For information on configuring Product management permissions for a Catalog, see [Creating and configuring Catalogs](#).

---

### Procedure

To work with Plans in a Catalog, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.  
The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.
2. If the Product that you want to work with is contained within a Space, select the required Space by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.The Products page of the Space opens, and all of the Products available in that Space are displayed.
3. Alongside the Product version that you want to work with, click the options icon  and then click View Plans.  
A window opens listing all the Plans in the selected Product.
4. Alongside the Plan that you want to work with, click the options icon .  
You can then select from the following options:
  - View Product: a window opens listing the Product that contains the Plan. If required, you can click the options icon  to work with the Product. For details of the available options, see [Managing your Products](#).
  - View APIs: a window opens listing the APIs in the Plan. If required, you can click the options icon  alongside an API to work with that API. For details of the available options, see [Working with APIs](#).
  - View Subscriptions: a window opens listing the subscriptions to the Plan. If required, you can click the options icon  alongside a subscription to work with that subscriptions. For details of the available options, see [Working with application subscriptions](#).

---

## Working with consumer organizations

Manage the consumer organizations that access your APIs and Plans when their users sign up to use the IBM® API Connect Developer Portal.

---

### About this task

Application developers in consumer organizations access the Developer Portal and sign up to use the Plans in the Products you create in the API Manager. You can create more than one consumer organization and each organization has one owner. Application developers are added through the Developer Portal UI.

You can also create groups of consumer organizations. A consumer organization group is an efficient way of grouping application developers together to publish Products to, making it easier to control who has access to the Plans in the Products.

Use the following tasks to administer your consumer organizations and consumer organization groups:

- [Creating a consumer organization](#)  
If you have permission to manage developers, you can create new consumer organizations in a Catalog. consumer organizations contain the application developers that subscribe to use the APIs that are published to the Catalog
- [Editing a consumer organization](#)  
You can change the title of an IBM API Connect consumer organization, and also transfer ownership to a new organization owner.

- [Deleting a consumer organization](#)  
When you delete a consumer organization, all of the resources in that organization are removed from the cloud. However, the account of the organization owner remains in IBM API Connect.
- [Working with the applications in a consumer organization](#)  
You can work with the developer applications that have been registered in a consumer organization.
- [Working with the subscriptions in a consumer organization](#)  
You can work with the Plan subscriptions that have been created by the developer applications registered in a consumer organization.
- [Working with consumer organization groups](#)  
A *consumer organization group* is a collection of consumer organizations, and provides an efficient way of controlling who can see, and subscribe to, the APIs in your Products. By using a consumer organization group, you can define this access for all the developers in the organizations in that group in a single operation, rather than having to define access for the organizations separately.
- [Sending messages to consumer organization owners](#)  
You can send email messages to the owners of consumer organizations. You can also style the email to match your business theme.

---

## Creating a consumer organization

If you have permission to manage developers, you can create new consumer organizations in a Catalog. consumer organizations contain the application developers that subscribe to use the APIs that are published to the Catalog

### Before you begin

Your IBM® API Connect Developer Portal must be enabled to perform this task. For more information, see [Creating and configuring Catalogs](#). To complete this task, you must be defined as a member in a Catalog or Space (if Spaces are enabled), and you must be assigned a role that has the Consumer-Org\_ Manage permission. For more information, see [Adding provider organization users and assigning roles](#) and [Managing Catalog membership](#).


### About this task

You can create a consumer organization in either of the following ways:

- Invite a user to be a consumer organization owner, in which case they receive an invitation email with an activation link, and can complete the creation of the consumer organization and specify the title.
- Create the consumer organization yourself, specifying the owner and title. There are two ways you can specify the owner:
  - Specify the user ID of an existing user.
  - If you are using a Local User Registry, specify a new user ID; the user is added to the registry and becomes the owner of the consumer organization.

If Spaces are enabled in your Catalog, when you create a consumer organization, it is added to the Catalog and its Spaces. For more information about enabling Spaces, see [Using syndication in API Connect](#).

### Procedure

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
2. Optional: If Spaces are enabled in the Catalog, select the Space that you want to work with by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.  
For more information about enabling Spaces, see [Using syndication in API Connect](#).
3. In the navigation pane of the API Manager UI, click the Consumers tab.
4. To invite a user to be a consumer organization owner, complete the following steps:
  - a. Click Add\_ Invite organization owner.
  - b. Enter the email address of the organization owner.
  - c. Click Invite.  
The new consumer organization is added to the list, and an email invitation is sent to the owner; by using the activation link, the owner specifies the title of the organization. The status is shown as *Pending* until the recipient of the email clicks the link in the email to complete the creation of their Developer Portal account, after which the status changes to *Enabled*.
5. To create the consumer organization yourself, complete the following steps:
  - a. Click Add\_ Create organization.
  - b. Enter the organization Title. A Name is entered automatically.  
The value in the Name field is a single string that is used to identify the consumer organization in developer toolkit CLI commands. The title is used for display.  
To view the CLI commands to manage consumer organizations, see [apic consumer-orgs](#).
  - c. Select the required user registry.
  - d. Specify the organization owner; complete one or other of the following steps, either to specify an existing user, or to create a new user:
    - To specify the user ID of an existing API Connect user who is in the selected user registry, complete the following steps:
      - For the Type of user, select Existing.
      - In the Username field, enter the user ID of the organization owner  
After the consumer organization is created, the specified user can immediately log in to the Developer Portal associated with the Catalog.
    - If you selected a Local User Registry, then to specify a new user ID, complete the following steps:
      - For the Type of user, select New User.
      - In the Username field, enter the new user ID.
      - Enter the Email, First Name, Last Name, and Password.  
After the consumer organization is created, the new user will have been added to the selected registry and can immediately log in to the Developer Portal associated with the Catalog, by using the specified user ID and password.
  - e. Click Create to complete the creation of the consumer organization.

---

## Editing a consumer organization

You can change the title of an IBM® API Connect consumer organization, and also transfer ownership to a new organization owner.

### Before you begin

---

To complete this task, you must be defined as a member in a Catalog or Space (if Spaces are enabled), and you must be assigned a role that has the Consumer-Org\_2 Manage permission. For more information, see [Adding provider organization users and assigning roles](#) and [Managing Catalog membership](#).

### About this task

---


If you transfer ownership, the new organization owner can be an existing user, or you can create a new user. When the ownership changes, the new owner is assigned administration privileges for the organization, and the previous owner has their administration privileges removed. If required, you can restore the privileges to the previous owner as described in [Administering members and roles](#).

If Spaces are enabled in your Catalog, when you edit the details of a consumer organization, the updates are applied to the Catalog and its Spaces. For more information about enabling Spaces, see [Using syndication in API Connect](#).

### Procedure

---

To change the title of a consumer organization, or to transfer ownership to a new organization owner, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
2. Optional: If Spaces are enabled in the Catalog, select the Space that you want to work with by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.For more information about enabling Spaces, see [Using syndication in API Connect](#).
3. In the navigation pane of the API Manager UI, click the Consumers tab.
4. Click the name of the consumer organization that you want to work with.
5. To change the title, enter a new title in the Title field, then click Save.
6. To transfer ownership, select the user registry for the consumer organization owner you want to transfer ownership to.

The remaining procedure varies according to the type of the selected user registry, as follows:

- Local User Registry
  - Select whether the user is an Existing user or a New User.
    - For an existing user, complete the following steps:
      - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account.
      - Click Save.
    - For a new user, complete the following steps:
      - Enter a unique user name for the new user.
      - Supply an email address, name details, and a password.
      - Click Save.
- LDAP
  - Enter the name of a user that exists in the selected user registry.
  - Click Save.
- Authentication URL and OIDC
  - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account. If the required organization owner has not previously been invited and activated their account, you must ensure they have been invited to the organization first.
  - Click Save.

### Results

---

If the title of the consumer organization is changed, all application developers that are members of that consumer organization will see the title change reflected in the Developer Portal user interface.

### Related tasks

---

- [Publishing a Product](#)

---

## Deleting a consumer organization

When you delete a consumer organization, all of the resources in that organization are removed from the cloud. However, the account of the organization owner remains in IBM® API Connect.

### Before you begin

---

To complete this task, you must be defined as a member in a Catalog or Space (if Spaces are enabled), and you must be assigned a role that has the Consumer-Org\_2 Manage permission. For more information, see [Adding provider organization users and assigning roles](#) and [Managing Catalog membership](#).

Important: If you delete a consumer organization, client IDs and secrets associated with applications that were registered by users in the consumer organization can no longer be used to call APIs. However, the account of the organization owner remains in API Connect.

## About this task



---

When you delete a consumer organization, it is removed permanently and users can no longer access that organization. If Spaces are enabled in your Catalog, when you delete a consumer organization, it is deleted from the Catalog and its Spaces. For more information about enabling Spaces, see [Using syndication in API Connect](#).

## Procedure

---

To delete a consumer organization, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
2. Optional: If Spaces are enabled in the Catalog, select the Space that you want to work with by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.For more information about enabling Spaces, see [Using syndication in API Connect](#).
3. In the navigation pane of the API Manager UI, click the Consumers tab.
4. Click the options icon  alongside the consumer organization you want to work with, then click Delete.
5. Click Confirm to delete the consumer organization.

Note: If the consumer organization is the only one specified in the custom visibility settings for a published Product, the deletion operation will fail. For more information, see [Changing the availability of a Product](#).

## Working with the applications in a consumer organization




---

You can work with the developer applications that have been registered in a consumer organization.

## Procedure

---

To work with applications, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
2. Optional: If Spaces are enabled in the Catalog, select the Space that you want to work with by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.For more information about enabling Spaces, see [Using syndication in API Connect](#).
3. In the navigation pane of the API Manager UI, click the Consumers tab.
4. Click the options icon  for the consumer organization that you want to work with, then click View Applications. A window opens listing all the developer applications that have been registered in the consumer organization.
5. To view and edit the details of an application, click the title of the application.
6. Further options are available by clicking the options icon  alongside the application you want to work with. For details, see [Working with developer applications](#).

## Working with the subscriptions in a consumer organization




---

You can work with the Plan subscriptions that have been created by the developer applications registered in a consumer organization.

## Procedure

---

To work with Plan subscriptions, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
2. Optional: If Spaces are enabled in the Catalog, select the Space that you want to work with by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.For more information about enabling Spaces, see [Using syndication in API Connect](#).
3. In the navigation pane of the API Manager UI, click the Consumers tab.
4. Click the options icon  for the consumer organization that you want to work with, then click View Subscriptions. A window opens listing all the Plan subscriptions that have been created by the developer applications registered in the consumer organization. If required, you can click the options icon  alongside a subscription to work with that subscription. For details of the available options, see [Working with application subscriptions](#).

## Working with consumer organization groups

---

A *consumer organization group* is a collection of consumer organizations, and provides an efficient way of controlling who can see, and subscribe to, the APIs in your Products. By using a consumer organization group, you can define this access for all the developers in the organizations in that group in a single operation, rather than having to define access for the organizations separately.

## About this task

---

A consumer organization group might represent particular business partners, internal organizations, or other groups of application developers.

When you create a Product, you can define which application developers can access the APIs in that Product. You can configure the following types of access:



- Visibility, which specifies the application developers who can see the APIs in the Developer Portal.
- Subscribability, which specifies the application developers who can create subscribe to use the APIs.



You configure this access by selecting the required consumer organizations, and Consumer organization groups. You can also change the visibility and subscribability settings for a Product after it has been published. For more information, see [Creating a draft Product](#) and [Changing the availability of a Product](#).

## Procedure

---

To work with consumer organization groups, you first navigate to the Consumer organization groups page. You then create new groups, send an email to the owners of the consumer organizations in a group, and edit or delete existing groups.

- Navigating to the required consumer organization groups page.
  1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
  2. If Spaces are enabled in the Catalog, select the Space that you want to work with by completing the following steps:
    - a. Click the Spaces tab.
    - b. Select the Space that you want to work with.
  3. Click the Consumers tab, then click Manage Groups.
- Creating a consumer organization group.
  1. Click Add.
  2. Enter a Title for the new group and, optionally, a Summary. A Name is entered automatically. The title is used for display.
  3. Use the Search orgs/groups field to search for, and select, the consumer organizations that you want to add to the group.  
Note: To prevent excessive search results lists, only the first 10 matches are displayed when you begin typing in the field; lengthen your search string to shorten the list for locating the required consumer organization.
  4. Click Save to create the group.
- Editing a consumer organization group.
  1. Click the options icon  for the consumer organization group that you want to work with, then click Edit.
  2. Make the required updates, then click Save.

Note: If you remove a consumer organization from a group, and that group is specified in the custom visibility settings for a Product, applications in that consumer organization can no longer subscribe to the Product. However, any existing subscriptions are not affected. For information on managing visibility and subscribability settings, see [Editing a draft Product](#).
- Deleting a consumer organization group.
  1. Click the options icon  for the consumer organization group that you want to work with, then click Delete.
  2. Click Confirm to delete the group.
- Sending an email to the owners of the consumer organizations in a consumer group
  1. Click the options icon  for the consumer organization group that you want to work with, then click Message owners.  
The Send message window opens, and the email addresses of the consumer organization owners are displayed as the recipients.
  2. Enter the subject of the message in the Subject field.
  3. Enter the content of the message in the Message field, then click Send.

---

## Sending messages to consumer organization owners

You can send email messages to the owners of consumer organizations. You can also style the email to match your business theme.

### Before you begin

---

To complete this task, you must be defined as a member in a Catalog or Space (if Spaces are enabled), and you must be assigned a role that has the Consumer-Org\_> Manage permission. For more information, see [Adding provider organization users and assigning roles](#) and [Managing Catalog membership](#).


Email messages contain a sender name and address that's based on a hierarchical search of the configured sender details in API Connect. Depending on the email being sent, the search can start at the Space level, and then go through the Catalog, Provider Organization, and the Cloud Manager levels. The sender details that are used will be the first set of configured details found during this hierarchical search. To view which search path API Connect takes when looking for the sender details for each notification template, see tbc.



Note: You can also send email messages by using the developer toolkit CLI, or by using the API Connect REST APIs. See [API development and management commands](#), or [API Connect REST APIs](#), for more information.

## Procedure

---

To send a message, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
2. Optional: If Spaces are enabled in the Catalog, select the Space that you want to work with by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.

For more information about enabling Spaces, see [Using syndication in API Connect](#).
3. In the navigation pane of the API Manager UI, click the Consumers tab.
4. You can either send a message to the owner of a specific consumer organization, or to the owners of multiple selected consumer organizations.
  - To send a message to the owner of a specific consumer organization, click the options icon  alongside the consumer organization you want to work with, then click Message owner. The Send Message window opens, and the email address of the consumer organization owner is displayed as the recipient.
  - To send a message to the owners of multiple selected consumer organizations, complete the following steps:
    - Click the options icon  alongside the Add button, then click Message owners.

- Select the required consumer organizations, then click Next. The Send Message window opens, and the email addresses of the selected consumer organization owners are displayed as the recipients.
5. Enter the subject of the message in the Subject field.
  6. Select the Content type that you want to use for the entering the content of the message, from HTML, PlainText, or Both. The default content type is PlainText. An edit window for the selected content type is displayed, or both edit windows are displayed if Both is selected. For HTML content, only the tags and their attributes that are shown in the following table are allowed.

Table 1. List of allowed HTML tags and their attributes

HTML tag	Attribute
<a>	"class", "href", "hreflang", "style"
<em>	"class", "style"
<strong>	"class", "style"
<cite>	"class", "style"
<blockquote>	"class", "cite", "style"
<code>	"class", "style"
<ul>	"class", "type", "style"
<ol>	"class", "start", "type", "style"
<li>	"class", "style"
<dl>	"class", "style"
<dt>	"class", "style"
<dd>	"class", "style"
<h1>	"class", "id", "style"
<h2>	"class", "id", "style"
<h3>	"class", "id", "style"
<h4>	"class", "id", "style"
<h5>	"class", "id", "style"
<h6>	"class", "id", "style"
<p>	"class", "style"
<div>	"class", "style"
 	"class", "style"
<span>	"class", "style"
<img>	"class", "src", "alt", "data-entity-type", "data-entity-uuid", "data-align", "data-caption", "width", "height", "style"
<table>	"class", "id", "style"
<tr>	"class", "id", "style"
<td>	"class", "id", "style"

If an HTML tag that is not allowed is used in a notification, the tag and its contents are displayed in the email as plain text.

Images can be used by adding a  tag in the template. The src attribute for the image must be a fully qualified web URL, and must be externally accessible so that the email recipients can access the image. It's not possible to reference local images, they must be fully qualified URLs. It's also not possible to embed or attach images or other files in the emails.


7. Enter the content of the message.
8. Click Send.




## Working with developer applications

You can work with the applications that have been registered in the Developer Portal associated with a Catalog.

### Procedure

To work with developer applications, complete the following steps.

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.
2. Optional: If Spaces are enabled in the Catalog, select the Space that you want to work with by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.
For more information about enabling Spaces, see [Using syndication in API Connect](#).
3. Click the Applications tab.

All applications that have been registered in the Developer Portal associated with this Catalog are listed.
4. To view and edit the details of an application, click the title of the application.
5. Further options are available by clicking the options icon  alongside the application you want to work with, as follows:
  - Client ID: lists the Client IDs for the application. You can add new client ID/client secret credentials; you can copy the client secret only at creation time. An option is provided for deleting any of the credentials from the application.
  - Create Subscription: subscribe the application to a Plan in a Product; select the required Plan and Product, then click Create Subscription.
  - View Subscriptions: a window opens listing summary details of all the Plan subscriptions for the application. If required, you can click the options icon  alongside a subscription to work with that subscriptions. For details of the available options, see [Working with application subscriptions](#).
  - View Consumer Organization: a window opens listing the consumer organization that the application belongs to. If required, you can click the options icon  to work with the consumer organization. For details of the available options, see [Working with consumer organizations](#).







## Working with application subscriptions

You can work with the subscriptions to the Plans in all the Products in a specific Catalog. Plan subscriptions are created by application developers in the Developer Portal.

## Procedure

---

To work with application subscriptions, complete the following steps:

1. In the navigation pane of the API Manager UI, click  Manage, then select the Catalog that you want to work with.  
The Products page of the Catalog opens, and all of the Products available in that Catalog are displayed.
2. Optional: If Spaces are enabled in the Catalog, select the Space that you want to work with by completing the following steps:
  - a. Click the Spaces tab.
  - b. Select the Space that you want to work with.For more information about enabling Spaces, see [Using syndication in API Connect](#).
3. Click the Subscriptions tab.  
A window opens listing all subscriptions to the Plans in all the Products in the Catalog.
4. Alongside the subscription that you want to work with, click the options icon .  
You can then select from the following options:
  - View Consumer Organization: a window opens listing the consumer organization that the subscribing application belongs to. If required, you can click the options icon  to work with the consumer organization. For details of the available options, see [Working with consumer organizations](#).
  - View Application: a window opens listing the subscribing application. If required, you can click the options icon  to work with the application. For details of the available options, see [Working with developer applications](#).
  - View Plan: a window opens listing the Plan that the application is subscribed to. If required, you can click the options icon  to work with the Plan. For details of the available options, see [Working with Plans](#).
  - View Product: a window opens listing the Product that contains the Plan that the application is subscribed to. If required, you can click the options icon  to work with the Product. For details of the available options, see [Managing your Products](#).

---

## Security and authentication

In API Manager, you can use TLS profiles to secure the transmission of data between the management server and other API Connect subsystems and external services, and also configure user registries to securely authenticate your Catalogs and APIs.

For details of authentication in IBM API Connect, see the following subtopics:

- [Creating a TLS client profile](#)  
In the IBM API Connect API Manager interface, TLS profiles are used to secure transmission of data between the management server and other API Connect subsystems and external services. TLS and SSL certificates guarantee that information you submit will not be stolen or tampered with. In this topic, you learn how to create a TLS profile in API Manager.
- [Authenticating by using your enterprise user registry](#)  
IBM API Connect supports a variety of user registry types for authenticating users and securing APIs.
- [Configuring a native OAuth provider](#)  
Native OAuth providers are configured and managed by you within your cloud.
- [Configuring a third-party OAuth provider](#)  
Enter the secure endpoints to provide OAuth authentication from a third party.
- [OAuth concepts for API Connect](#)  
OAuth is a token-based authorization protocol that allows third-party websites or applications to access user data without requiring the user to share personal information.

---

## Creating a TLS client profile

In the IBM® API Connect API Manager interface, TLS profiles are used to secure transmission of data between the management server and other API Connect subsystems and external services. TLS and SSL certificates guarantee that information you submit will not be stolen or tampered with. In this topic, you learn how to create a TLS profile in API Manager.

---

## Before you begin

One of the following roles is required to configure TLS Profiles:

- Organization Administrator
- Owner
- Custom role with the `Settings: Manage permissions`

---

## About this task


API Connect supports the use of TLS and SSL certificates, but does not itself produce strong encryption keys or manage your encryption keys. Encryption keys should be created and managed according to your own procedures. For more information, see [Viewing certificate details and adding certificates to a keystore or truststore](#) and [Generating a PKCS#12 file for Certificate Authority](#).

Note: If you update a TLS profile that is associated with a Gateway service, the updates are not automatically propagated to Gateway servers. For instructions on configuring the toolkit command-line tool to use TLS certificates when connecting to API Manager, see [Configuring the command-line tool to use TLS certificates](#).

---

## Procedure

To create a TLS profile, complete the following steps:

1. In the API Manager, click  Resources.
2. Select TLS.
3. Click Create in the TLS Client Profile table.
4. Enter the fields to configure the TLS Client Profile:

Field	Description
Title (required)	Enter a Title for the profile. The title is displayed on the screen.
Name (required)	<p>The Name is auto-generated. The value in the Name field is a single string that can be used in developer toolkit CLI commands. To view the CLI commands to manage a TLS Client Profile, see <a href="#">apic tls-client-profiles</a>.</p> <p>Important: The name of the TLS Client Profile as saved on the DataPower® Gateway, depending on the gateway type, is as follows:</p> <ul style="list-style-type: none"> <li>• DataPower API Gateway: <code>provider-org-name_catalog-name_tlsp-tls-profile-nameV1.0.0</code></li> <li>• DataPower Gateway (v5 compatible): <code>provider-org-name-tls-profile-nameV1.0.0</code></li> </ul> <p>where</p> <ul style="list-style-type: none"> <li>• <code>tls-profile-name</code> is the value of the auto-generated Name field for the TLS client profile in API Connect.</li> <li>• <code>provider-org-name</code> is the name of the provider organization containing the TLS client profile.</li> <li>• <code>catalog-name</code> is the name of the Catalog, in that provider organization, containing the TLS client profile.</li> </ul>
Version (required)	Assign a version number for the profile. Using version numbers allows you to create multiple server profiles with the same name and different configurations, for example, <i>MyProfile 1.0</i> and <i>MyProfile 1.1</i> .
Summary (optional)	Enter a description of the profile.
Protocols (required)	Select one or more supported TLS protocol versions. The default is 1.2.
Server Connection (optional)	<p>Specify whether to support weak or insecure credentials.</p> <ul style="list-style-type: none"> <li>• Allow insecure server connections - Insecure server connections may result from self-signed certificates, expired or corrupted certificates, or certificates from an unknown or untrusted source. Check this box to allow the connection to proceed with an insecure connection. The default is to not allow insecure server connections.</li> <li>• Support Server Name Indication (SNI) - Check this box to enable SNI. SNI allows support for multiple certificates presented on the same IP address using different host names. The client profile sends the name of a virtual domain as part of the TLS negotiation. The default is to enable SNI.</li> </ul>
Keystore (optional)	A Keystore is a repository containing public and private key pairs. Select the keystore where you will store the certificates for the profile. Default keystores are provided, and you can also create your own.
Truststore (optional)	A Truststore is a repository containing verified public keys, which are usually obtained from a third-party certificate authority. Truststores provide list of certificates to verify a peer's certificate. If used in a TLSServerProfile, a truststore is used when mutual authentication is enabled. Select a truststore for the profile. Default truststores are provided, and you can also create your own.
Ciphers (required)	<p>Cipher suites are encryption/decryption algorithms used to secure HTTPs communication within the API Connect ecosystem. Select the ciphers that the profile supports.</p> <p>Note: The TLS 1.3 ciphers are clearly indicated. If you select TLS version 1.3 as one of the protocols for the profile but do not select any TLS 1.3 ciphers, <b>all</b> the TLS 1.3 ciphers are added to the list of ciphers supported by the profile. If you do not select TLS version 1.3 but select one or more TLS 1.3 ciphers, those ciphers are <b>not</b> added to the list of ciphers supported by the profile.</p>

5. Click Save.

- [Creating a Keystore](#)  
Keystores contain matched pairs of public certificates and private keys used to confirm identity and encrypt/decrypt data transmission over HTTPS.
- [Creating a Truststore](#)  
Truststores are repositories containing trusted certificates with verified public keys. The certificates in the truststore are usually obtained from a third-party certificate authority (CA).
- [Generating a PKCS#12 file for Certificate Authority](#)  
PKCS#12 (P12) files define an archive file format for storing cryptographic objects as a single file. API Connect supports the P12 file format for uploading a keystore and truststore. The keystore should contain both a private and public key along with intermediate CA certificates.
- [Generating a self-signed certificate using OpenSSL](#)  
OpenSSL is an open source implementation of the SSL and TLS protocols. It provides the transport layer security over the normal communications layer, allowing it to be intertwined with many network applications and services.
- [Viewing certificate details and adding certificates to a keystore or truststore](#)  
You can view details for the certificates in an existing keystore or truststore and add additional certificates. You might need add certificates when a certificate or PKCS #12 (P12) file expires.
- [Defining elliptic curve cryptographic schemes for a TLS client profile](#)  
You define the elliptic curve cryptographic schemes for a TLS client profile by using the developer toolkit CLI.

## Related tasks

- [Working with user registries](#)

## Creating a Keystore

Keystores contain matched pairs of public certificates and private keys used to confirm identity and encrypt/decrypt data transmission over HTTPS.

## Before you begin



API Manager supports and uses TLS certificates, but does not produce strong encryption keys or manage your encryption keys. Encryption keys are generated and managed according to your own procedures. For more information, see [Generating a PKCS#12 file for Certificate Authority](#) and [Generating a self-signed certificate using OpenSSL](#).

One of the following roles is required to configure Keystores:


- Organization Administrator
- Owner
- Custom role with the `Settings: Manage permissions`

## About this task

API Connect includes pre-configured Keystores which may be used for testing purposes. For production environments, we suggest creating a new, secure Keystore.

## Procedure

Perform the following steps to create a TLS Client profile:

1. In the API Manager, click  Resources.
2. Select TLS.
3. Click Create in the Keystore table.

Field	Description
Title (required)	Enter a Title for the Keystore. The title is displayed on the screen.
Name (required)	The Name is auto-generated. The value in the Name field is a single string that can be used in developer toolkit CLI commands. To view the CLI commands to manage keystores, see <a href="#">apic keystores</a> .
Summary (optional)	Enter a brief description.
Private Key & Public Key: Step 1: Upload private key	Upload the file containing the private key certificate. If necessary, you can click Browse to locate the file. If the file contains both the private and public keys, upload it in Step 1. Private and public keys are always uploaded in pairs, either in a single file or separate files.
Private key password (optional)	Enter the password for the private key if it has a password.
Private Key & Public Key: Step 2: Upload public key	If the public key is contained in a separate file, upload it in Step 2. Private and Public keys are always uploaded in pairs, either in a single file or separate files.

4. Click Save.

Note: After they have been uploaded, private keys cannot be downloaded from API Connect.

## Related tasks

- [Creating a TLS client profile](#)
- [Creating a Truststore](#)
- [Generating a PKCS#12 file for Certificate Authority](#)

## Creating a Truststore

Truststores are repositories containing trusted certificates with verified public keys. The certificates in the truststore are usually obtained from a third-party certificate authority (CA).

## Before you begin

One of the following roles is required to configure truststores:


- Organization Administrator
- Owner
- Custom role with the `Settings: Manage permissions`

## About this task

API Manager supports and uses TLS certificates, but does not produce strong encryption keys or manage your encryption keys. Encryption keys are generated and managed according to your own procedures. For more information, see [Generating a PKCS#12 file for Certificate Authority](#) and [Generating a self-signed certificate using OpenSSL](#).

API Connect includes pre-configured Truststores which may be used for testing purposes. For production environments, we suggest creating a new, secure Truststore.

## Procedure

1. In the API Manager, click  Resources.
2. Select TLS.
3. Click Create in the Truststore table.

Field	Description
Title (required)	Enter a Title for the Truststore. The title is displayed on the screen.
Name (required)	The Name is auto-generated. The value in the Name field is a single string that can be used in developer toolkit CLI commands. To view the CLI commands to manage truststores, see <a href="#">apic truststores</a> .

Field	
Summary (optional)	Enter a brief description.
Public Keys	Upload the file containing the public key certificate. If necessary you can click Browse to locate the file.

4. Click Save.

## Related tasks

- [Creating a TLS client profile](#)
- [Creating a Keystore](#)

## Generating a PKCS#12 file for Certificate Authority

PKCS#12 (P12) files define an archive file format for storing cryptographic objects as a single file. API Connect supports the P12 file format for uploading a keystore and truststore. The keystore should contain both a private and public key along with intermediate CA certificates.

### Before you begin

One of the following roles is required to add a key to a keystore or truststore:

- Organization Administrator
- Owner
- Custom role with the `Settings: Manage permissions`

Before you can generate a P12 file, you must have a private key (for example: `key.pem`), a signed certificate by a Certificate Authority (for example `certificate.pem`) and one or more certificates from the CA authority (known as *intermediate CA certificates*).

Note: If your certificate file contains more than one certificate, you must manually split the file and create a single file for each entry. Each entry must be bound by the following markers:

```
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
```

### Procedure

1. If you have intermediate certificates from your CA, concatenate them into a single `.pem` file to build your `caChain`. Be sure to enter a new line following each certificate's data.


```
cat ca1.pem ca2.pem ca3.pem > caChain.pem
cat caChain.pem
-----BEGIN CERTIFICATE-----
MIIEPjCCA46gAwIBAgIQEod26KZabjd+BQMG1Dw16jANBgkqhkiG9w0BAQUFADCBC
...
lQX7CkTJn61AJUsyEa8H/gjVQnHp4VOLFR/dKgeVcCRvZF7Tt5AuiyHY
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIEPDCCAySgAwIBAgIQSEus8arH1xND0aJ0NUmXJTANBgkqhkiG9w0BAQUFADBVB
...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIEAjCCAx6gAwIBAgIBATANBgkqhkiG9w0BAQUFADBvMQswCQYDVQQGEWJTRTEU
...
-----END CERTIFICATE-----
```

2. Create the P12 file including the private key, the signed certificate and the CA file you created in step 1, if applicable. Omit the `-CAfile` option if you don't have CA certificates to include.

The following command uses OpenSSL, an open source implementation of the SSL and TLS protocols.

```
openssl pkcs12 -inkey key.pem -in certificate.pem -export -out certificate.p12 -CAfile caChain.pem -chain
```

Once the certificate file is created, it can be uploaded to a keystore.

3. Once the certificate file is created, it can be uploaded to a Truststore. In the API Manager, click .
4. Select TLS.
5. Click Create in the Keystore table.
6. Create a Keystore and upload the certificate file following the instructions at [Creating a Keystore](#).

Note:

- API Connect supports only the P12 (PKCS12) format file for the present certificate.
- Your P12 file must contain the private key, the public certificate from the Certificate Authority, and all intermediate certificates used for signing.
- Your P12 file can contain a maximum of 10 intermediate certificates.

7. Click Save.

### Results

Your P12 file is generated and ready to upload.

### What to do next

Upload your P12 file to IBM® API Connect. For more information, see [Creating a TLS client profile](#).

---

## Generating a self-signed certificate using OpenSSL

OpenSSL is an open source implementation of the SSL and TLS protocols. It provides the transport layer security over the normal communications layer, allowing it to be intertwined with many network applications and services.

---

### Before you begin

One of the following roles is required to complete this task:

- Organization Administrator
- Owner
- Custom role with the `Settings: Manage permissions`

---

### About this task

This topic tells you how to generate a self-signed SSL certificate request using the OpenSSL toolkit to enable HTTPS connections.

---

### Procedure

To generate a self-signed SSL certificate using the OpenSSL, complete the following steps:

1. Write down the Common Name (CN) for your SSL Certificate. The CN is the fully qualified name for the system that uses the certificate. For static DNS, use the hostname or IP address set in your Gateway Cluster (for example, `192.16.183.131` or `dp1.acme.com`).
2. Run the following OpenSSL command to generate your private key and public certificate. Answer the questions and enter the Common Name when prompted.

```
openssl req -newkey rsa:2048 -nodes -keyout key.pem -x509 -days 365 -out certificate.pem
```

3. Review the created certificate:

```
openssl x509 -text -noout -in certificate.pem
```


4. Combine your key and certificate in a PKCS#12 (P12) bundle:

```
openssl pkcs12 -inkey key.pem -in certificate.pem -export -out certificate.p12
```

5. Validate your P2 file.

```
openssl pkcs12 -in certificate.p12 -noout -info
```

Once the certificate file is created, it can be uploaded to a keystore.

6. In the API Manager, click  Resources.
7. Select TLS.
8. Click Create in the Keystore table.
9. Create a Keystore and upload the certificate file following the instructions at [Creating a Keystore](#).

Note:

- API Connect supports only the P12 (PKCS12) format file for the present certificate.
- Your P12 file must contain the private key, the public certificate from the Certificate Authority, and all intermediate certificates used for signing.
- Your P12 file can contain a maximum of 10 intermediate certificates.

10. Click Save.

---

### Related tasks

- [Creating a TLS client profile](#)
- [Creating a Keystore](#)
- [Creating a Truststore](#)

---

## Viewing certificate details and adding certificates to a keystore or truststore

You can view details for the certificates in an existing keystore or truststore and add additional certificates. You might need add certificates when a certificate or PKCS #12 (P12) file expires.

---

### Before you begin

One of the following roles is required to edit Keystores and Truststores:

- Organization Administrator
- Owner
- Custom role with the `Settings: Manage permissions`


---

### About this task

API Connect includes pre-configured Keystores and Truststores which can be used for testing purposes. For production environments, we suggest creating a new, secure Keystore or Truststore. From an existing Keystore or Truststore, you can edit the title, view the details of the certificates, and add new certificates.

## Procedure

To work with existing Keystores and Truststores:

1. In the API Manager, click  Resources.
2. Select TLS.
3. Click the name of the keystore or truststore.
4. Edit the name and summary if needed.
5. The Subject, Finger Print, and Expiration date is shown for each certificate in the keystore or truststore. Click > to view the certificate details.
6. Add additional private and/or public keys if needed.
7. Click Save.

Note: After they have been uploaded, private keys cannot be downloaded from API Connect.

## Related tasks

- [Creating a Truststore](#)
- [Creating a Keystore](#)
- [Generating a PKCS#12 file for Certificate Authority](#)
- [Generating a self-signed certificate using OpenSSL](#)

## Defining elliptic curve cryptographic schemes for a TLS client profile

You define the elliptic curve cryptographic schemes for a TLS client profile by using the developer toolkit CLI.

### About this task

To define elliptic curve cryptographic schemes for a TLS client profile, you include `elliptic_curve_auto_negotiation` and `elliptic_curve` properties in a YAML file definition for the TLS client profile. The `elliptic_curve` property lists the required elliptic curve cryptographic schemes. For example:

```
elliptic_curve_auto_negotiation: false
elliptic_curve:
 - secp521r1
 - secp384r1
 - prime256v1
```

You then use the developer toolkit CLI to create the TLS client profile in API Connect.

When `elliptic_curve_auto_negotiation` is set to `true`, the system negotiates the Elliptic-curve Diffie-Hellman (ECDH) key agreement automatically with its peer, and any `elliptic_curve` property settings are ignored.

The following example shows a complete YAML file for a TLS client profile:

```
type: tls_client_profile
name: my-tls-client-profile
version: 1.0.0
title: My TLS client profile
protocols:
 - tls_v1.2
ciphers:
 - ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
elliptic_curve_auto_negotiation: false
elliptic_curve:
 - sect163k1
insecure_server_connections: false
server_name_indication: true
```

Note:

- The `elliptic_curve_auto_negotiation` option is not supported by any of the API Connect gateway types. If the TLS client profile is targeted for an API Connect gateway; this setting is ignored by the gateway.
- The elliptic curve cryptographic schemes shown in each row of the following table are equivalent. However, API Connect recognizes only one or the other depending on how the TLS profile is used, as indicated in the table.

Table 1.

API enforcement on the gateway	API Connect server access security
secp192r1	prime192v1
secp256r1	prime256v1

Therefore, if you want to use either of these schemes and are unsure whether you are targeting the TLS client profile to the API Connect gateway for API enforcement, whether you are using it to secure user access to the API Connect servers, or whether it will be used for both purposes, specify both equivalent schemes; API Connect will simply ignore the non-relevant scheme. For example:

```
elliptic_curve:
 .
 .
 .
 - secp256r1
 - prime256v1
 .
 .
```

## Procedure

---

To create a TLS client profile with elliptic curve cryptographic schemes defined, complete the following steps:

1. Create a YAML file definition for your TLS client profile, with the required `elliptic_curve` property.
2. Log in to the management server from the developer toolkit CLI. Log in either as a member of the cloud administration organization or as a member of a provider organization, depending on where you want to create the TLS client profile. For details, see [Logging in to a management server](#).
3. Create the TLS client profile by using the following command:

```
apic tls-client-profiles:create --server mgmt_endpoint_url --org organization_name tls_client_profile_yaml_file
```

where:

- `mgmt_endpoint_url` is the platform API endpoint URL, and is the same as that which was used when you logged in at step 2.
- `organization_name` is either `admin`, for the cloud administration organization, or the name of your provider organization, and is the same as that which was used when you logged in at step 2.
- `tls_client_profile_yaml_file` is the name of the YAML file that contains the definition for your TLS client profile.

Note: When you install IBM® API Connect, the API Connect gateway has a pre-supplied default TLS client profile that is used for API enforcement if you do not configure a TLS client profile; you cannot configure this default TLS client profile on the gateway.

For reference details of all the `apic tls-client-profiles` commands, see [apic tls-client-profiles](#).

You can also complete the operations described in this topic by using the API Connect REST APIs; see the [API Connect REST API documentation](#).

---

## Authenticating by using your enterprise user registry

IBM® API Connect supports a variety of user registry types for authenticating users and securing APIs.

You can use your enterprise user registry for authentication in API Connect if it is of one of the following types:

LDAP directory

If your user registry uses Lightweight Directory Access Protocol (LDAP), you can use it in API Connect for both user authentication and API security.

Authentication URL User Registry

You can configure a non-LDAP user registry by using an authentication URL user registry. An authentication URL user registry enables integration with third-party authentication providers. You can use an authentication URL user registry in API Connect for both user authentication and API security.

Local User Registry

You can authenticate users with a local user registry. A local user registry is an internal registry stored within API Connect.

OpenID Connect

You can authenticate users through an identity provider that supports the OpenID Connect (OIDC) authentication protocol.

- [Working with user registries](#)  
To secure the APIs that are published to your IBM API Connect Catalogs, you authenticate with user registries.
- [Modifying the configuration details for a user registry](#)  
You modify the configuration details for a user registry by using the User Registries page in the API Manager user interface of IBM API Connect.
- [Password lockout criteria](#)  
You can be locked out of your account if you attempt to log in and fail consecutively.

## Related tasks

---

- [Working with user registries](#)

---

## Working with user registries

To secure the APIs that are published to your IBM® API Connect Catalogs, you authenticate with user registries.

## About this task


---

Create user registry resources to secure your APIs, and authenticate with users.

## Procedure

---

To configure a user registry, complete the following steps:

1. In the API Manager user interface, select  Resources.
2. On the Resources page, select User Registries > Create.
3. Select the tile for the user registry type you want, and continue to the instructions for configuring user authentication for your selection.  
Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.

Authentication URL User Registry

For more information, see [Creating an Authentication URL user registry](#).

Custom user registry

For more information, see [Creating an organization-specific custom user registry](#).

- LDAP User Registry  
For more information, see [Creating an LDAP user registry in API Manager](#).
- Local User Registry  
For more information, see [Creating a Local User Registry](#)
- OIDC User Registry  
For more information, see [Creating an OIDC user registry](#).

## Results

---

The user registry information is added to API Manager.

## What to do next

---

You can now use API Manager to chart and manage your Catalogs.

- [Creating an Authentication URL user registry](#)  
An Authentication URL user registry provides a simple mechanism for authenticating users by referencing a custom identity provider.
- [Creating an organization-specific custom user registry](#)  
You can configure an organization-specific custom user registry to provide user authentication for the Developer Portal.
- [Creating an LDAP user registry in API Manager](#)  
You can use the API Manager UI to configure an organization-specific LDAP user registry to provide user authentication and onboarding for the Developer Portal. APIs can also be secured with an LDAP user registry.
- [Creating a Local User Registry](#)  
A Local User Registry (LUR) can be created to provide user authentication for API Manager.
- [Creating an OIDC user registry](#)  
Create an organization-specific OIDC user registry when multi-factor authentication (MFA) is required.

---

## Creating an Authentication URL user registry

An Authentication URL user registry provides a simple mechanism for authenticating users by referencing a custom identity provider.

## About this task

---

This topic describes how to create a new Authentication URL user registry as a Resource in your organization. After the user registry is created, the user registry must be added to the Sandbox catalog.

One of the following roles is required to configure user registries:

- Administrator
- Organization Owner
- Custom role with the `Settings: Manage permissions`

Note:

API Connect issues an HTTP `GET` call to the Authentication URL endpoint, sending the user's credential. The following example shows a call made to an Authentication URL identity provider with an endpoint defined as `https://myauthurl.example.com/user/authenticate`:

```
GET /user/authenticate HTTP/1.1
Host: myauthurl.example.com
Authorization: Basic c3Bvb246Zm9yaw=
```

If the Authentication URL endpoint returns an HTTP status code of `200`, the user authenticates successfully. An HTTP status code other than `200` indicates a failed login attempt. API Connect forwards any HTTP Header starting with `X-` (with the exception of `X-Client-Certificate`), and `Cookie` to the Authentication URL identity provider, to aid the authentication decision; for example:

```
GET /user/authenticate HTTP/1.1
Host: myauthurl.example.com
Authorization: Basic c3Bvb246Zm9yaw=
X-Forwarded-For: 8.8.9.9
X-Custom-Header-From-Customer: special
Cookie: MyCookie=VGhpc01zV21ja2VkQW1hemluZw==
```

When a user is presented with the form for completing their API Connect user registration, which fields are pre-populated depends on which fields are returned in the response from the Authentication URL identity provider. If any of the following fields are returned, they will be pre-populated in the registration form:

- `username`
- `email`
- `first_name`
- `last_name`

If the `username` field is not returned, the registration form displays the user name that was provided by the user. The pre-population capability requires that the response from the Authentication URL identity provider satisfies the following conditions:


- The `Content-Type` must be `application/json`.
- The response body format must be JSON.

A sample response is as follows:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
```

```
{
 "username": "myuser",
 "email": "myuser@example.com",
 "first_name": "My",
 "last_name": "User"
}
```

## Procedure

1. In the API Manager, click  Resources.
2. Select User Registries to see the list of current user registries in your organization.
3. Click Create in the User Registries section.

Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.

4. Select Authentication URL User Registry and enter the following parameters:

Field	Description
Title (required)	Enter a descriptive name to use on the screen.
Name (required)	The name that is used in CLI commands. The name is auto-generated. For details of the CLI commands for managing user registries, see <a href="#">apic user-registries</a> .
Summary (optional)	Enter a brief description.
Display Name (required)	The name that is displayed for selection by the user when logging in to a user interface, or activating their API Manager account. For details of user interface log in, and account activation, see <a href="#">Accessing the Cloud Manager user interface</a> , <a href="#">Accessing the API Manager user interface</a> , and <a href="#">Activating your API Manager user account</a> .  Note: The Developer Portal uses the <b>Title</b> of the User Registries when rendering them at the login page, rather than the <b>Display Name</b> .
URL (required)	Enter the URL for the authentication service. When establishing authentication, API Connect makes a GET call to the URL. The call includes the user name and password it has collected from the user in its authorization header. Either 200 OK or 401 Unauthorized will be returned.
TLS Client Profile (optional)	Select a TLS Client Profile to allow secure authentication with a specific web server.
Case sensitive	To ensure proper handling of user name capitalization, you <b>must</b> ensure that your case-sensitivity setting here matches the setting on your backend Authentication URL server: <ul style="list-style-type: none"> <li>• <b>Only</b> select Case sensitive if your backend server supports case-sensitivity.</li> <li>• Do <b>not</b> select Case sensitive if your backend server does not support case-sensitivity.</li> </ul> Note: The Developer Portal does not support case sensitive usernames. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
Email required	Select this checkbox if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this checkbox if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

5. Click Save.
6. Add the user registry to the Sandbox catalog. See [Creating and configuring Catalogs](#).

## Results

The user registry can be used for Basic Authentication in the Security Definition for an API. For more information, see [Creating a basic authentication security definition](#).

## Creating an organization-specific custom user registry

You can configure an organization-specific custom user registry to provide user authentication for the Developer Portal.

### Before you begin

Custom user registries can be used for authenticating users to the Developer Portal, but cannot be used to secure APIs.

To configure a custom user registry as a resource in API Manager, the external user directory must be created and available to use with your API Connect ecosystem.

Also, a custom user registry definition for your external user registry must already have been created on the Cloud Manager, as you will need the information from this definition to set up the custom user registry. See [Configuring a shared custom user registry](#) for information.

One of the following roles is required to configure a custom user registry:

- Administrator
- Owner
- Topology Administrator
- Custom role with the Settings: Manage permissions

## About this task

You can create a custom user registry that is specific to a provider organization, or one that can be shared and available to all of the provider organizations in your API Connect environment. An organization-specific user registry can be used for authenticating Developer Portal users in a specific provider organization. While a shared user registry can be used for authenticating Cloud Manager, API Manager, and Developer Portal users.

This topic describes how to configure an organization-specific custom user registry. If you want to create a shared registry, see [Configuring a shared custom user registry](#) for more information.

Note:

- You can also create and manage custom user registries by using the API Connect REST APIs; see the [API Connect REST API documentation](#).

You can use the following instructions to create a writable or a read-only custom user registry.

API Connect provides two methods for creating a custom user registry in the API Manager, as described in the following sections:

- [Creating a custom user registry by using the API Manager UI](#)
- [Creating a custom user registry by using the developer toolkit CLI](#)

Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.

## Creating a custom user registry by using the API Manager UI

You configure a custom user registry by creating a new custom user registry resource, which references the custom user registry type that is defined in the Cloud Manager. To make the custom user registry available to the Developer Portal for user authentication, you must enable the registry in the associated Catalog. When the custom user registry is used for authentication, API Connect makes a REST call to the endpoint of your external registry, as defined in the custom user registry type.

## Creating a custom user registry resource

Use the following instructions to create a new custom user registry resource in the API Manager UI.

- Click Create in the User Registries section of Resources.
- Click the Custom user registry tile.
- Enter the following information:

Property	Description
Custom type	The name of the custom user registry type. For example, <i>my-custom-user-registry-type</i> . Use the drop-down arrow to select different custom user registry types. The custom user registry type is defined in the Cloud Manager.
Title	A descriptive name to display in the UI.
Name	The name of the custom user registry. This name is auto-generated, and is used in CLI commands.
Summary	A brief description of the custom user registry.
Endpoint	Optional endpoint information.
Case sensitive	To ensure proper handling of user name capitalization, you <b>must</b> ensure that your case-sensitivity setting here matches the setting on your backend server: <ul style="list-style-type: none"><li><b>Only</b> select Case sensitive if your backend server supports case-sensitivity.</li><li>Do <b>not</b> select Case sensitive if your backend server does not support case-sensitivity.</li></ul> Note: <ul style="list-style-type: none"><li>The Developer Portal does not support case sensitive usernames.</li><li>After at least one user has been onboarded into the registry, you cannot change this setting.</li></ul>
User registry managed	Determines whether API Connect manages your user registry. Valid values are: <ul style="list-style-type: none"><li><b>true</b> - select the checkbox</li><li><b>false</b> - clear the checkbox</li></ul>
User managed	Determines whether your user registry is writable or not. Select the checkbox to set to <b>true</b> for writable. Clear the checkbox for the non-writable option.
Email required	Select this checkbox if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this checkbox if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

- Click Create. The new user registry is displayed in the User registries list.

## Adding your custom user registry to the Developer Portal login

To make the custom user registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. In the API Manager UI, click Manage followed by the relevant Catalog, and then click Catalog settings > Onboarding. In the Catalog User Registries section, click Edit, select the custom user registry, and click Save. For more information, see [Creating and configuring Catalogs](#).

## Creating a custom user registry by using the developer toolkit CLI

You configure an organization-specific custom user registry by creating a new custom user registry resource that references the integration document that exists on the Cloud Manager. You use developer toolkit CLI commands to create the custom user registry, and to make the registry available to the Developer Portal, you must enable the registry in the associated Catalog. When the custom user registry is used for authentication, API Connect makes a REST call to the endpoint of your external registry, as defined in the integration document.



## Logging in to the management server CLI

Before you can define the custom user registry configuration, you must log in to your management server from the developer toolkit CLI as a member of a provider organization. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details of the login command, see [Logging in to a management server](#).

For more information about how to use the CLI, see [Installing the toolkit](#), and [Overview of the command-line tool](#).

## Defining your custom user registry

You define the configuration of your custom user registry in a `custom_config_file.yaml` file, as shown in the following example.

```
name: 'custom_registry_name'
title: 'display_title'
registry_type: 'custom_user_registry'
integration_url: custom_integration_url
case_sensitive: true_or_false
user_managed: true_or_false
user_registry_managed: true_or_false
email_required: true_or_false
email_unique_if_exist: true_or_false
identity_providers:
- name: provider_name
 title: 'provider_title'
configuration:
 custom_config1: 'value1'
 customize: true
```

The registry properties that are common to each authentication method are described in the following table:

Property	Description
<code>name</code>	The name of the custom user registry. This name is used in the CLI commands.
<code>title</code>	A descriptive name to display in a graphical user interface.
<code>registry_type</code>	The registry type that is configured in the <code>name</code> property in the integration document, for example <code>custom_user_registry</code> .
<code>integration_url</code>	The custom integration URL in your API Connect configuration. You can determine the custom integration URL by using the following CLI command: <pre>apic integrations:list --server <i>mgmt_endpoint_url</i> --subcollection <i>user-registry</i></pre>
<code>case_sensitive</code>	Determines whether your user registry is case-sensitive. Valid values are: <ul style="list-style-type: none"><li><code>true</code></li><li><code>false</code></li></ul> To ensure proper handling of user name capitalization, you <b>must</b> ensure that your case-sensitivity setting here matches the setting on your backend LDAP server: <ul style="list-style-type: none"><li><b>Only</b> set <code>case_sensitive</code> to <code>true</code> if your backend LDAP server supports case-sensitivity.</li><li>Set <code>case_sensitive</code> to <code>false</code> if your backend LDAP server does not support case-sensitivity.</li></ul> Note: After at least one user has been onboarded into the registry, you cannot change this setting.
<code>user_managed</code>	Determines whether your user registry is writable or not. Must be set to <code>true</code> for writable. Set to <code>false</code> if you don't want the registry to be writable.
<code>user_registry_managed</code>	Determines whether API Connect manages your user registry. Valid values are: <ul style="list-style-type: none"><li><code>true</code></li><li><code>false</code></li></ul>
<code>email_required</code>	Determines whether an email address is required as part of the user onboarding process. Valid values are: <ul style="list-style-type: none"><li><code>true</code></li><li><code>false</code></li></ul> If set to <code>true</code> , the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.

Property	Description
<code>email_unique_if_exists</code>	Determines whether email addresses must be unique within the user registry. Valid values are: <ul style="list-style-type: none"> <li><code>true</code></li> <li><code>false</code></li> </ul> <p>Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.</p>
<code>identity_providers</code>	An array containing the details of your custom server, where: <ul style="list-style-type: none"> <li><code>name</code> - is the name of the custom server and is the name that is used in CLI commands</li> <li><code>title</code> - is the display name of the custom server</li> </ul>
<code>configuration</code>	The user-defined configuration based on the <code>configuration_schema</code> that is defined in the integration document.

Save your `custom_config_file.yaml` so it can be accessed by the `user-registries:create` command in the following section.

## Creating your custom user registry

To create your organization-specific custom user registry, run the following CLI command:

```
apic user-registries:create --server mgmt_endpoint_url --org organization_name custom_config_file.yaml
```

where:

- `mgmt_endpoint_url` is the platform API endpoint URL.
- `--org organization_name` means that the registry will be created in your provider organization.
- `custom_config_file` is the name of the YAML file that defines the configuration of your custom user registry.

On completion of the registry creation, the command displays the following summary details:

```
registry_name registry_url
```

The `registry_name` is derived from the `name` property in the custom user registry YAML file. The `registry_url` is the URL with which the custom registry resource can be accessed.

Your organization-specific user registry is now created; see the following section for instructions on how to make the registry available to users.

## Configuring your custom user registry in a Catalog

To make your custom user registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. Complete the following steps:

- Determine the URL of your custom user registry by using the following command (or you can copy and paste from the summary of the registry creation):

```
apic user-registries:list --server mgmt_endpoint_url --org organization_name
```

- Enter the following command (the terminating hyphen character means that the command takes input from the command line):

```
apic configured-catalog-user-registries:create --server mgmt_endpoint_url --org organization_name --catalog catalog_name -
```

where `catalog_name` is the value of the `name` property of the required Catalog. The command returns

```
Reading CONFIGURED_CATALOG_USER_REGISTRY_FILE arg from stdin
```

- Enter the following data, followed by a new line:

```
user_registry_url: custom_registry_url
```

where `custom_registry_url` is the URL of your custom user registry, obtained in step [1](#).

- Press **CTRL D** to terminate the input.

## Related information

- [Command-line tool reference for the developer toolkit](#)

## Creating an LDAP user registry in API Manager

You can use the API Manager UI to configure an organization-specific LDAP user registry to provide user authentication and onboarding for the Developer Portal. APIs can also be secured with an LDAP user registry.

## Before you begin

To configure an LDAP user registry as a resource in API Manager, the LDAP directory must be created and populated for use with your API Connect ecosystem.

LDAP registries can be used to secure APIs, or for securing a Catalog to authenticate Developer Portal users.

Important: If you are using an LDAP registry to secure APIs, the STARTTLS protocol, which upgrades an insecure protocol to a secure one by applying TLS security, is not supported.

One of the following roles is required to configure an LDAP user registry:

- Organization Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage permissions`

## About this task

You can create an LDAP user registry that is specific to a provider organization, or one that can be shared and available to all of the provider organizations in your API Connect environment. An organization-specific LDAP user registry can be used for authenticating Developer Portal users in a specific provider organization. While a shared LDAP user registry can be used across the Cloud Manager, the API Manager, and the Developer Portal components in your environment.

This topic describes how to configure an organization-specific LDAP user registry. If you want to create a shared registry, see [Configuring an LDAP user registry in the Cloud Manager](#) for more information.

Note:

- You can also create and manage LDAP user registries by using the developer toolkit CLI. For more information, see [Using the CLI to create an organization-specific LDAP user registry](#).
- The API Connect REST APIs can also be used to create and manage LDAP user registries; see the [API Connect REST API documentation](#).
- If you are using the DataPower® API Gateway, LDAP group authentication is not supported.
- You can map external LDAP groups to API Connect user roles to enable greater control of user access, but this configuration can be done only by using the developer toolkit CLI; see [Using the CLI to create an organization-specific LDAP user registry](#) for details.


You create an LDAP user registry by configuring a set of properties in the API Manager UI. If you want to enable writable LDAP, you must complete the Attribute Mapping section by selecting the User Managed checkbox, and providing the mapping of your source LDAP attribute names to the target API Connect values. You can also change a registry to be read-only again by clearing the User Managed checkbox. To make the registry available to the Developer Portal, you must define the registry for consumer onboarding in the associated Catalog. To secure APIs with an LDAP registry, you must configure security definitions.

For general information about authenticating with LDAP, see [LDAP authentication](#).

## Procedure

Follow these steps to configure a new LDAP user registry as a Resource in the API Manager UI.

Note: If you are using an **Active Directory**, you must indicate this by using the property `"directory_type": "ad"` in the LDAP config.

1. In the API Manager, click  Resources.
2. Click Create in the User Registries section.

**Important:** Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.

3. Select LDAP User Registry for the user registry type, and enter the following information:

Field	Description
Title	Enter a descriptive name to display on the screen.
Name	The name that is used in CLI commands. The name is auto-generated. For details of the CLI commands for managing user registries, see <a href="#">apic user-registries</a> .
Display Name (required)	The name that is displayed for selection by the user when logging in to a user interface, or activating their API Manager account. For details of user interface log in, and account activation, see <a href="#">Accessing the Cloud Manager user interface</a> , <a href="#">Accessing the API Manager user interface</a> , and <a href="#">Activating your API Manager user account</a> .  Note: The Developer Portal uses the <b>Title</b> of the User Registries when rendering them at the login page, rather than the <b>Display Name</b> .
Summary (optional)	Enter a brief description.
Address	Enter the IP address or host name of the LDAP server.
Port	Enter the Port number that API Connect can use to communicate with the LDAP registry. For example, 389.
Select a TLS Client Profile (optional)	Select the TLS Client Profile that the LDAP server requires.
Select a protocol version	Select the version number for the LDAP protocol that you are using.
Case sensitive	To ensure proper handling of user name capitalization, you <b>must</b> ensure that your case-sensitivity setting here matches the setting on your backend LDAP server: <ul style="list-style-type: none"> <li>• <b>Only</b> select Case sensitive if your backend LDAP server supports case-sensitivity.</li> <li>• <b>Do not</b> select Case sensitive if your backend LDAP server does not support case-sensitivity.</li> </ul> Note: The Developer Portal does not support case sensitive usernames. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
Email required	Select this checkbox if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this checkbox if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

4. Click Next and enter the authentication information, which will vary depending on the selected Authentication Method. The choices are:
  - **Compose DN** - Select this format if you can compose the user LDAP Distinguished Name (DN) from the user name. For example, `uid=<username>,ou=People,dc=company,dc=com` is a DN format that can be composed from the user name. If you are unsure whether Compose (DN) is the correct option, contact your LDAP administrator. If you are using an LDAP registry to secure APIs, Compose DN is not supported with the DataPower API Gateway.

- Compose UPN - Select this format if your LDAP directory supports binding with User Principal Names such as `john@acme.com`. The Microsoft Active Directory is an example of an LDAP directory that supports Compose UPN authentication. If you are unsure whether your LDAP directory supports binding with UPNs, contact your LDAP administrator.  
Note: The Admin Bind DN and Admin Bind Password are not used with this authentication method.
- Search DN - Select this format if you cannot compose the user LDAP Distinguished Name from the user name; for example, if the base DNs of the users are different. This format might require an administrator DN and password to search for users in the LDAP directory. If your LDAP directory permits anonymous binds, you can omit the admin DN and password. If you are unsure if your LDAP directory permits anonymous binds, contact your LDAP administrator.

For all of the authentication methods:

If you are creating an LDAP registry to authenticate users of an API, you can specify an LDAP authorization group to restrict API access. To be able to call an API that is secured by the LDAP registry, a user must successfully authenticate with their LDAP user ID and password **and** they must be a member of the specified authorization group. The authorization group can be a Static Group or Dynamic Group. A static group is one in which the individual members of the group are explicitly listed. A dynamic group is one which is defined according to the set of attributes that the group members share in common.

5. For authentication method Compose DN, enter the following:

Field	Description
Bind Method	Anonymous or Authenticated. If specific permissions are not needed to search the registry, select Anonymous Bind. Or, if specific permissions are necessary, select Authenticated Bind.
Admin DN	For Authenticated Bind, enter the Distinguished Name of a user authorized to perform searches in the LDAP directory. For example <code>cn=admin,dc=company,dc=com</code> .
Admin Password	For Authenticated Bind, enter the user password for the Admin DN.
Prefix	Specify the prefix to the DN. For example <code>(uid=</code> .
Suffix	Specify the suffix to the DN. For example <code>)</code> .
Base DN (optional)	Enter a base DN in the Base DN field, or click Get Base DN to populate the field with a retrieved base DN.
Use group authentication (optional)	Static or Dynamic. For Static Group, enter the Group Based DN, Prefix, and Suffix. For Dynamic Group, enter the Filter condition for the group.

6. For authentication method Compose UPN, enter the following:

Field	Description
Bind Method	Anonymous or Authenticated. If specific permissions are not needed to search the registry, select Anonymous Bind. Or, if specific permissions are necessary, select Authenticated Bind.
Admin DN	For Authenticated Bind, enter the Distinguished Name of a user authorized to perform searches in the LDAP directory. For example <code>cn=admin,dc=company,dc=com</code> .
Admin Password	For Authenticated Bind, enter the user password for the Admin DN.
Suffix	Enter the domain part of the user principal name. For example, <code>@acme.com</code> .
Use group authentication (optional)	Enter the Filter condition for the group.

7. For authentication method Search DN, enter the following:

Field	Description
Bind Method	Anonymous or Authenticated. If specific permissions are not needed to search the registry, select Anonymous Bind. Or, if specific permissions are necessary, select Authenticated Bind.
Admin DN	For Authenticated Bind, enter the Distinguished Name of a user authorized to perform searches in the LDAP directory. For example <code>cn=admin,dc=company,dc=com</code> .
Admin Password	For Authenticated Bind, enter the user password for the Admin DN.
Prefix	Specify the prefix to the DN. For example <code>(uid=</code> .
Suffix	Specify the suffix to the DN. For example <code>)</code> .
Base DN (optional)	Enter a base DN in the Base DN field, or click Get Base DN to populate the field with a retrieved base DN.
Use group authentication (optional)	Static or Dynamic. For Static Group, enter the Group Based DN, Prefix, and Suffix. For Dynamic Group, enter the Filter condition for the group.

8. Optional: Click Test configuration to test the settings for your LDAP user registry. Enter valid credentials to ensure that you can access the LDAP database.

9. Optional: If you want to make your LDAP user registry writable, select the User Managed checkbox in the Attribute Mapping section, and provide the mapping of your source LDAP attribute names to the target API Connect values. Click Add to add each name/value pair, specified as follows:

- LDAP ATTRIBUTE NAME - is the name of the source LDAP attribute.
- API CONNECT VALUE - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in [ ] with the value that the user supplies when signing up.

The default user profile properties that API Connect requires during user registration are `username`, `first_name`, `last_name`, `email`, and `password`, as shown in the following example:

LDAP ATTRIBUTE NAME	API CONNECT VALUE
<code>dn</code>	<code>uid=[username],ou=users,dc=company,dc=com</code>
<code>cn</code>	<code>[first_name] [last_name]</code>
<code>sn</code>	<code>[last_name]</code>
<code>mail</code>	<code>[email]</code>
<code>userPassword</code>	<code>[password]</code>

You must ensure that you enter the correct attribute mapping values for your LDAP configuration, to enable API Connect to access the LDAP database. Note that a writable LDAP user registry cannot be used to authenticate Cloud Manager and API Manager users.

10. Click Create.

Your new LDAP registry is shown in the list of User Registries on the Resources page.

## What to do next

If you want to make the LDAP user registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. Click the relevant Catalog, then click Settings, Onboarding. In the Catalog User Registries section, click Edit, select the user registry, and click Save. For more information, see [Creating and configuring Catalogs](#).

If you want to use the LDAP user registry to secure APIs, see the following information:

- To use for basic authentication in the security definition for an API, see [Creating a basic authentication security definition](#).

- To use for authentication in the User Security configuration for a native OAuth provider, see [Configuring user security for a native OAuth provider](#).
- **Using the CLI to create an organization-specific LDAP user registry**  
You can use the developer toolkit CLI to configure an organization-specific LDAP user registry to provide user authentication for the Developer Portal. APIs can also be secured with an LDAP user registry.

## Related information

- [Command-line tool reference for the developer toolkit](#)
- [Securing your API Connect Cloud with LDAP \(series of developer articles\)](#)

## Using the CLI to create an organization-specific LDAP user registry

You can use the developer toolkit CLI to configure an organization-specific LDAP user registry to provide user authentication for the Developer Portal. APIs can also be secured with an LDAP user registry.

### Before you begin

To configure an LDAP user registry as a resource in API Manager, the LDAP directory must be created and available to use with your API Connect ecosystem.

LDAP registries can be used to secure APIs, or for securing a Catalog to authenticate Developer Portal users.

Important: If you are using an LDAP registry to secure APIs, The STARTTLS protocol, which upgrades an insecure protocol to a secure one by applying TLS security, is not supported.

Note: If you are using an **Active Directory**, you must indicate this by using the property `"directory_type": "ad"` in the LDAP config. One of the following roles is required to configure an LDAP user registry:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage permissions`

### About this task

You can create an LDAP user registry that is specific to a provider organization, or one that can be shared and available to all of the provider organizations in your API Connect environment. An organization-specific LDAP user registry can be used for onboarding and authenticating Developer Portal users in a specific provider organization. While a shared LDAP user registry can be used for authenticating Cloud Manager, API Manager, and Developer Portal users.

This topic describes how to configure an organization-specific LDAP user registry. If you want to create a shared registry, see [Using the CLI to configure a shared LDAP user registry](#) for more information.

Note:

- You can also create organization-specific LDAP user registries by using the API Manager UI; for more information see [Creating an LDAP user registry in API Manager](#).
- In addition, you can create and manage LDAP user registries by using the API Connect REST APIs; see the [API Connect REST API documentation](#).
- If you are using the DataPower® API Gateway, LDAP group authentication is not supported.
- You can map external LDAP groups to API Connect user roles to enable more control of user authorization by using the developer toolkit CLI. The following instructions explain how to set the `external_group_mapping_enabled` configuration on your LDAP user registry resource. For information about how to set external role mapping on your API Connect user roles, see [Configuring LDAP group mapping on Cloud Manager user roles](#).

You create an LDAP user registry by first defining the registry details in a configuration file. You then use a developer toolkit CLI command to create the registry, passing the configuration file as a parameter. To make the registry available to the Developer Portal, you must enable the registry in the associated Catalog. To secure APIs with an LDAP registry, you must configure security definitions. You can use the following instructions to create a writable or a read-only LDAP user registry.

For general information about authenticating with LDAP, see [LDAP authentication](#).

## Logging in to the management server CLI

Before you can define the LDAP user registry configuration, you must log in to your management server from the developer toolkit CLI as a member of a provider organization. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`. For full details of the login command, see [Logging in to a management server](#).

For more information about how to use the CLI, see [Installing the toolkit](#), and [Overview of the command-line tool](#).

## Defining your LDAP configuration

You define the configuration of your LDAP user registry in an `ldap_config_file.yaml` file, as shown in the following example. Note that the actual contents of your YAML file will vary depending on the authentication method of your LDAP server, and this is explained in the following tables. Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.

```
name: registry_name
title: "display_title"
integration_url: LDAP_integration_url
user_managed: true_or_false
user_registry_managed: false

external_group_mapping_enabled: true_or_false
case_sensitive: true_or_false
email_required: true_or_false
email_unique_if_exist: true_or_false
identity_providers:
- name: provider_name
 title: provider_title
endpoint:
 endpoint: "ldap_server_url_and_port"
configuration:
 authentication_method: authentication_method
 authenticated_bind: "true_or_false"
 admin_dn: "admin_dn"
 admin_password: admin_password
 search_dn_base: "search_dn_base"
 search_dn_scope: search_dn_scope
 search_dn_filter_prefix: prefix
 search_dn_filter_suffix: suffix
 attribute_mapping:
 dn: "distinguished_name"
 cn: "common_name"
 sn: "last_name"
 mail: "email_address"
 userPassword: "password"
```

The registry properties that are common to each authentication method are described in the following table:

Property	Description
<code>name</code>	The name of the registry. This name is used in CLI commands.
<code>title</code>	A descriptive name to display in a graphical user interface.
<code>integration_url</code>	The LDAP integration URL in your API Connect configuration. You can determine the LDAP integration URL by using the following CLI command: <code>apic integrations:list --server mgmt_endpoint_url --subcollection user-registry</code>
<code>user_managed</code>	Determines whether your user registry is writable or not. Must be set to <code>true</code> for writable LDAP. You can change this setting to <code>false</code> if you don't want the registry to be writable; see the Switching your LDAP registry between writable and read-only section at the end of this topic for details. Note that a writable LDAP user registry cannot be used to authenticate Cloud Manager and API Manager users.
<code>user_registry_managed</code>	Must be set to <code>false</code> for LDAP. Determines whether API Connect manages your user registry. Only LUR registries are managed by API Connect.
<code>external_group_mapping_enabled</code>	Determines whether your user registry supports LDAP group mapping. Valid values are: <ul style="list-style-type: none"> <li><code>true</code></li> <li><code>false</code></li> </ul> The default value is <code>false</code> .
<code>case_sensitive</code>	Determines whether your user registry is case-sensitive. Valid values are: <ul style="list-style-type: none"> <li><code>true</code></li> <li><code>false</code></li> </ul> To ensure proper handling of user name capitalization, you <b>must</b> ensure that your case-sensitivity setting here matches the setting on your backend LDAP server: <ul style="list-style-type: none"> <li><b>Only</b> set <code>case_sensitive</code> to <code>true</code> if your backend LDAP server supports case-sensitivity.</li> <li>Set <code>case_sensitive</code> to <code>false</code> if your backend LDAP server does not support case-sensitivity.</li> </ul> Note: After at least one user has been onboarded into the registry, you cannot change this setting.
<code>email_required</code>	Determines whether an email address is required as part of the user onboarding process. Valid values are: <ul style="list-style-type: none"> <li><code>true</code></li> <li><code>false</code></li> </ul> If set to <code>true</code> , the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.

Property	Description
<code>email_unique_if_exist</code>	Determines whether email addresses must be unique within the user registry. Valid values are: <ul style="list-style-type: none"> <li><code>true</code></li> <li><code>false</code></li> </ul> <p>Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.</p>
<code>identity_providers</code>	An array containing the details of your LDAP server, where: <ul style="list-style-type: none"> <li><code>name</code> - is the name of the LDAP server and is the name that is used in CLI commands</li> <li><code>title</code> - is the display name of the LDAP server</li> </ul>
<code>endpoint</code>	The endpoint of your LDAP server, made up of the url and port, for example: <code>"ldap://server.com:389"</code>
<code>tls_profile</code>	Optionally set the TLS Client Profile that the LDAP server requires.
<code>protocol_version</code>	Optionally set the version number for the LDAP protocol that you are using. Valid values are: <ul style="list-style-type: none"> <li>2</li> <li>3</li> </ul> <p>Defaults to 3 if not explicitly set.</p>

The properties in the configuration section will vary depending on the selected authentication method. The three authentication methods are:

- `compose_dn`** - Set this format if you can compose the user LDAP Distinguished Name (DN) from the user name. For example, `uid=<username>,ou=People,dc=company,dc=com` is a DN format that can be composed from the user name. If you are unsure whether Compose (DN) is the correct option, contact your LDAP administrator. If you are using an LDAP registry to secure APIs, `compose_dn` is not supported with the DataPower API Gateway.
- `compose_upn`** - Set this format if your LDAP directory supports binding with User Principal Names such as `john@acme.com`. The Microsoft Active Directory is an example of an LDAP directory that supports Compose UPN authentication. If you are unsure whether your LDAP directory supports binding with UPNs, contact your LDAP administrator.  
Note: The Admin Bind DN and Admin Bind Password are not used with this authentication method.
- `search_dn`** - Select this format if you cannot compose the user LDAP Distinguished Name from the user name; for example, if the base DNs of the users are different. This format might require an administrator DN and password to search for users in the LDAP directory. If your LDAP directory permits anonymous binds, you can omit the admin DN and password. If you are unsure if your LDAP directory permits anonymous binds, contact your LDAP administrator.

For authentication method `compose_dn`, set the following configuration properties:

Properties	Description
<code>authentication_method</code>	<code>compose_dn</code>
<code>authenticated_bind</code>	The bind method. Valid values are: <ul style="list-style-type: none"> <li><code>"true"</code> - authenticated bind</li> <li><code>"false"</code> - anonymous bind</li> </ul> <p>If specific permissions are not needed to search the registry, select <code>"false"</code>. If specific permissions are necessary, select <code>"true"</code>.</p>
<code>admin_dn</code>	If <code>authenticated_bind</code> is set to <code>"true"</code> , enter the Distinguished Name (DN) of a user authorized to perform searches in the LDAP directory. For example: <code>"cn=admin,dc=company,dc=com"</code>
<code>admin_password</code>	If <code>authenticated_bind</code> is set to <code>"true"</code> , enter the user password for the <code>admin_dn</code> .
<code>search_dn_base</code>	Optionally set a base DN, for example: <code>"dc=company,dc=com"</code>
<code>bind_prefix</code>	Set the prefix to the DN, for example: <code>(uid=</code>
<code>bind_suffix</code>	Set the suffix to the DN, for example: <code>)</code>
<code>attribute_mapping</code>	If <code>user_managed</code> is set to <code>true</code> , provide the mapping of your source LDAP attribute names to the target API Connect values. This mapping is configured as a name/value pair, specified as follows: <code>ldap_registry_attribute_name: "apic_ldap_attribute_value"</code> Where: <ul style="list-style-type: none"> <li><code>ldap_registry_attribute_name</code> - is the name of the source LDAP attribute</li> <li><code>apic_ldap_attribute_value</code> - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in [ ] with the value that the user supplies when signing up.</li> </ul> <p>The user profile properties that API Connect requires during user registration are <code>username</code>, <code>first_name</code>, <code>last_name</code>, <code>email</code>, and <code>password</code>. The following extract shows an example of an attribute mapping:</p> <pre>attribute_mapping: dn: "uid=[username],ou=users,dc=company,dc=com" cn: "[first_name] [last_name]" sn: "[last_name]" mail: "[email]" userPassword: "[password]"</pre>

For authentication method `compose_upn`, set the following configuration properties:

Properties	Description
------------	-------------

Properties	Description
<code>authentication_method</code>	<code>compose_upn</code>
<code>authenticated_bind</code>	<p>The bind method. Valid values are:</p> <ul style="list-style-type: none"> <li>"true" - authenticated bind</li> <li>"false" - anonymous bind</li> </ul> <p>If specific permissions are not needed to search the registry, select "false". If specific permissions are necessary, select "true".</p>
<code>admin_dn</code>	<p>If <code>authenticated_bind</code> is set to "true", enter the Distinguished Name (DN) of a user authorized to perform searches in the LDAP directory. For example:</p> <p><code>"cn=admin,dc=company,dc=com"</code></p>
<code>admin_password</code>	<p>If <code>authenticated_bind</code> is set to "true", enter the user password for the <code>admin_dn</code>.</p>
<code>bind_suffix</code>	<p>Enter the domain part of the user principal name. For example:</p> <p><code>@acme.com</code></p>
<code>attribute_mapping</code>	<p>If <code>user_managed</code> is set to <code>true</code>, provide the mapping of your source LDAP attribute names to the target API Connect values. This mapping is configured as a name/value pair, specified as follows:</p> <p><code>ldap_registry_attribute_name: "apic_ldap_attribute_value"</code></p> <p>Where:</p> <ul style="list-style-type: none"> <li><code>ldap_registry_attribute_name</code> - is the name of the source LDAP attribute</li> <li><code>apic_ldap_attribute_value</code> - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in [ ] with the value that the user supplies when signing up.</li> </ul> <p>The user profile properties that API Connect requires during user registration are <code>username</code>, <code>first_name</code>, <code>last_name</code>, <code>email</code>, and <code>password</code>. The following extract shows an example of an attribute mapping:</p> <p>attribute_mapping:  dn: "uid=[username],ou=users,dc=company,dc=com"  cn: "[first_name] [last_name]"  sn: "[last_name]"  mail: "[email]"  userPassword: "[password]"</p>

For authentication method `search_dn`, set the following configuration properties:

Property	Description
<code>authentication_method</code>	<code>search_dn</code>
<code>authenticated_bind</code>	<p>The bind method. Valid values are:</p> <ul style="list-style-type: none"> <li>"true" - authenticated bind</li> <li>"false" - anonymous bind</li> </ul> <p>If specific permissions are not needed to search the registry, select "false". If specific permissions are necessary, select "true".</p>
<code>admin_dn</code>	<p>If <code>authenticated_bind</code> is set to "true", enter the Distinguished Name (DN) of a user authorized to perform searches in the LDAP directory. For example:</p> <p><code>"cn=admin,dc=company,dc=com"</code></p>
<code>admin_password</code>	<p>If <code>authenticated_bind</code> is set to "true", enter the user password for the <code>admin_dn</code>.</p>
<code>search_dn_base</code>	<p>Optionally set a base DN, for example:</p> <p><code>"dc=company,dc=com"</code></p>
<code>search_dn_scope</code>	<p>Optionally set the search DN scope. The scope determines which part of the directory information tree is examined. Possible values are:</p> <ul style="list-style-type: none"> <li>base</li> <li>one</li> <li>sub</li> </ul>
<code>search_dn_filter_prefix</code>	<p>Set the prefix to the DN, for example:</p> <p><code>(uid=</code></p>
<code>search_dn_filter_suffix</code>	<p>Set the suffix to the DN, for example:</p> <p><code>)</code></p>



Property	Description
<code>attribute_mapping</code>	<p>If <code>user_managed</code> is set to <code>true</code>, provide the mapping of your source LDAP attribute names to the target API Connect values. This mapping is configured as a name/value pair, specified as follows:</p> <pre>ldap_registry_attribute_name: "apic_ldap_attribute_value"</pre> <p>Where:</p> <ul style="list-style-type: none"> <li><code>ldap_registry_attribute_name</code> - is the name of the source LDAP attribute</li> <li><code>apic_ldap_attribute_value</code> - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in [ ] with the value that the user supplies when signing up.</li> </ul> <p>The user profile properties that API Connect requires during user registration are <code>username</code>, <code>first_name</code>, <code>last_name</code>, <code>email</code>, and <code>password</code>. The following extract shows an example of an attribute mapping:</p> <pre>attribute_mapping:   dn: "uid=[username],ou=users,dc=company,dc=com"   cn: "[first_name] [last_name]"   sn: "[last_name]"   mail: "[email]"   userPassword: "[password]"</pre>

Save your `ldap_config_file.yaml` so it can be accessed by the `user-registries:create` command in the following section. See the [Example](#) section for an example configuration file.

## Creating your LDAP user registry

To create your organization-specific LDAP user registry, run the following CLI command:

```
apic user-registries:create --server mgmt_endpoint_url --org organization_name ldap_config_file.yaml
```

where:

- `mgmt_endpoint_url` is the platform API endpoint URL.
- `organization_name` is the value of the `name` property of your provider organization.
- `ldap_config_file` is the name of the YAML file that defines the configuration of your LDAP user registry.

On completion of the registry creation, the command displays the following summary details:

```
registry_name registry_url
```

The `registry_name` is derived from the `name` property in the configuration YAML file. The `registry_url` is the URL with which the registry resource can be accessed. Your LDAP user registry is now created; see the following section for instructions on how to make the registry available in the Developer Portal.

## Configuring your LDAP registry in a Catalog

If you want to make your LDAP registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. Complete the following steps:

- Determine the URL of your LDAP user registry by using the following command (or you can copy and paste from the summary of the registry creation):

```
apic user-registries:list --server mgmt_endpoint_url --org organization_name
```

- Log in to the management server as a member of a provider organization; enter the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details of the `apic login` command, see [Logging in to a management server](#).

- Enter the following command (the terminating hyphen character means that the command takes input from the command line):

```
apic configured-catalog-user-registries:create --server mgmt_endpoint_url --org organization_name --catalog catalog_name -
```

where `catalog_name` is the value of the `name` property of the required Catalog. The command returns

```
Reading CONFIGURED_CATALOG_USER_REGISTRY_FILE arg from stdin
```

- Enter the following data, followed by a new line:

```
user_registry_url: ldap_registry_url
```

where `ldap_registry_url` is the URL of your LDAP registry, obtained in step [1](#).

5. Press **CTRL D** to terminate the input.

## Switching your LDAP registry between writable and read-only

---

After an LDAP user registry has been created, it can be switched between writable and read-only by updating the `user_managed` property in the registry configuration. Complete the following steps.

1. Determine the name or ID of the LDAP user registry that you want to update, by running the following command (or you can use the summary from the registry creation):

```
apic user-registries:list --server mgmt_endpoint_url --org organization_name
```

The command returns a list of all the user registries for that organization, shown by name followed by their registry URL. The registry ID is located at the end of the URL, for example `https://company.com/api/user-registries/x-x-x-x-x/registry_id`.

2. Enter the following command (the terminating hyphen character means that the command takes input from the command line):

```
apic user-registries:update --server mgmt_endpoint_url --org organization_name registry_name_or_id -
```

where `registry_name_or_id` is the name or ID of the LDAP user registry that you want to update (as determined in the previous step). The command returns:

```
Reading USER_REGISTRY_FILE arg from stdin
```

3. Enter the following data, followed by a new line:

```
user_managed: true_or_false
```

where `true` makes the registry writable, and `false` makes the registry read-only.

4. Press **CTRL D** to terminate the input.

Note that if you are changing your registry from read-only to writable, you must also set the `attribute_mapping` configuration, as described in the previous registry property tables.

## Using an LDAP user registry to secure APIs

---

If you want to use the LDAP user registry to secure APIs, see the following information:

- To use for basic authentication in the security definition for an API, see [Creating a basic authentication security definition](#).
- To use for authentication in the User Security configuration for a native OAuth provider, see [Configuring user security for a native OAuth provider](#).

For details of all the `apic user-registries` and `apic configured-catalog-user-registries` commands, see [apic user-registries](#) and [apic configured-catalog-user-registries](#).

## Example

---

The following example shows a configuration file that uses the Search DN authentication method for setting up writable LDAP:

```
name: sdn-ldap
title: "SDN LDAP User Registry"
integration_url: https://mycompany.com/api/cloud/integrations/user-registry/xxx-xxx-xxx
user_managed: true
user_registry_managed: false

case_sensitive: false
identity_providers:
- name: ldap
 title: "SDN LDAP Identity Provider"
endpoint:
 endpoint: "ldap://mycompany.com:389"
configuration:
 authentication_method: search_dn
 authenticated_bind: "true"
 admin_dn: "cn=admin,dc=company,dc=com"
 admin_password: xxxx
 search_dn_base: "dc=company,dc=com"
 search_dn_scope: sub
 search_dn_filter_prefix: (uid=
 search_dn_filter_suffix:)
 attribute_mapping:
 dn: "uid=[username],ou=users,dc=company,dc=com"
 cn: "[first_name] [last_name]"
 sn: "[last_name]"
 mail: "[email]"
 userPassword: "[password]"
```

## Related information

---

- [Command-line tool reference for the developer toolkit](#)
- [🔗 Securing your API Connect Cloud with LDAP \(series of developer articles\)](#)

## Creating a Local User Registry

---

A Local User Registry (LUR) can be created to provide user authentication for API Manager.

## About this task

Local User Registries (LURs) are the default user registries included in API Connect. LURs are local databases included with API Connect. Two default LURs are installed and configured during installation of API Connect. They cannot be deleted. The default Admin user account is stored in the Provider LUR.


You can use API Manager to create additional Local User Registries for use with your provider organization.

One of the following roles is required to configure user registries:

- Administrator
- Organization Owner
- Custom role with the `Settings: Manage permissions`

## Procedure

Follow these steps to configure a new LUR:

1. In the API Manager, click  Resources.
2. Select User Registries to see the list of current user registries in your organization.
3. Click Create in the User Registries section.

Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.

4. Select Local User Registry as the type for the user registry and enter the following information:

Field	Description
Title (required)	Enter a descriptive name for use on the screen.
Name (required)	The name that is used in CLI commands. The name is auto-generated. For details of the CLI commands for managing user registries, see <a href="#">apic user-registries</a> .
Display Name (required)	The name that is displayed for selection by the user when logging in to a user interface, or activating their API Manager account. For details of user interface log in, and account activation, see <a href="#">Accessing the Cloud Manager user interface</a> , <a href="#">Accessing the API Manager user interface</a> , and <a href="#">Activating your API Manager user account</a> .  Note: The Developer Portal uses the <b>Title</b> of the User Registries when rendering them at the login page, rather than the <b>Display Name</b> .
Summary (optional)	Enter a brief description.
Case sensitive	Select this setting if user names are case-sensitive. Note: The Developer Portal does not support case sensitive usernames. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
Email required	Select this checkbox if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this checkbox if email addresses must be unique within the user registry. For new Local User Registries, this setting is always selected; so if email addresses are contained in the user record, they must be unique. However, for existing Local User Registries this setting can be edited. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

5. Click Save.
6. Add the user registry to the Sandbox Catalog. See [Creating and configuring Catalogs](#).

## Results

The user registry can be used for Basic Authentication in the Security Definition for an API. For more information, see [Creating a basic authentication security definition](#).

## Creating an OIDC user registry

Create an organization-specific OIDC user registry when multi-factor authentication (MFA) is required.

You can create an OIDC user registry that is specific to a provider organization, or that is shared and available to all the provider organizations in your API Connect environment. An organization-specific OIDC user registry is used for onboarding and authenticating Developer Portal users, while a shared OIDC user registry can be used for onboarding and authenticating Cloud Manager, API Manager, and Developer Portal users.

This topic describes how to create an organization-specific registry. For information on how to create a shared registry, see [Configuring an OIDC user registry](#).


Important: If you are using Twitter as your OIDC provider, API Connect supports only the OAuth 1.0a method, so your Twitter application must be configured to use OAuth 1.0a. Other OIDC providers support OAuth 2.0.

API Connect provides two methods for creating an OIDC user registry in API Designer, as described in the following sections:

- [Using the UI to create an OIDC user registry](#)
- [Using the CLI to create an OIDC user registry](#)

## Using the UI to create an OIDC user registry

Use the API Designer user interface to create an organization-specific OIDC user registry when multi-factor authentication (MFA) is required.

1. In the API Designer navigation pane, click  Resources.
2. Click User Registries.
3. Click Create and select OIDC User Registry.

Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.

4. On the Create OIDC User Registry page, use the fields in each of the following sections to configure the registry settings, and then click Create. Many of the registry settings are preconfigured to simplify the configuration steps.

#### Provider Type

Use the settings in Table 1 to define the provider type.

Table 1. Provider Type settings

Field	Description
Provider Type	OIDC provider. Select one of the following supported OIDC providers: <ul style="list-style-type: none"> <li>• Facebook</li> <li>• GitHub</li> <li>• Google</li> <li>• LinkedIn</li> <li>• Slack</li> <li>• Twitter</li> <li>• Windows Live</li> <li>• Standard OIDC (default value allows you to specify another provider)</li> </ul>
Title	Provide a descriptive name for display purposes.
Name	Automatically generated. This name is used in CLI commands to reference the registry. For details of the CLI commands for managing user registries, see the <a href="#">apic user-registries</a> topic in the Command Line tool reference section of this documentation.
Summary	Provide a brief description of the new registry.
Email required	Select this check box if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this check box if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

#### Provider Endpoint

Automatically generated for most supported providers. In the Authorization Endpoint field, type the URL of the provider's authorization endpoint.

#### Token Endpoint

Fill in the settings as described in Table 2.

Table 2. Token Endpoint settings

Field	Description
URL	Preconfigured for most of the supported OIDC providers. Type the URL of the provider's token endpoint.
TLS	Select the TLS Client Profile for the token endpoint (OIDC must be configured to use TLS). Default is Default TLS Client Profile.

#### UserInfo Endpoint

Fill in the settings as described in Table 3.

Table 3. UserInfo Endpoint settings

Field	Description
URL	Preconfigured for most of the supported OIDC providers. Type the URL of the provider's userinfo endpoint.
TLS	Select the TLS Client Profile for the userinfo endpoint (OIDC must be configured to use TLS). Default is Default TLS Client Profile.

#### JWKS Endpoint

Fill in the settings as described in Table 4.

Table 4. JWKS Endpoint settings

Field	Description
URL	Type the URL of the read-only endpoint that contains the public keys' information in JWKS format.
TLS	Select the TLS Client Profile for the userinfo endpoint (OIDC must be configured to use TLS). Default is Default TLS Client Profile.

#### Logout

Fill in the settings as described in Table 5.

Table 5. Logout

Field	Description
Logout redirect	Optionally, give a logout redirect URL, allowing APIC to initiate an https 302 redirect to the configured endpoint. Note: When the Developer Portal uses this feature, it does not retain control afterwards. Instead, the content is presented from the redirected endpoint.

#### Client Information

Fill in the settings as described in Table 6.

Table 6. Client Information settings

Field	Description
Client ID	Provide the client ID of the application that is registered with the selected OIDC provider.
Client Secret	Provide the client secret of the application that is registered with the selected OIDC provider.
Response Type	Preconfigured for most of the supported OIDC providers. Specify the data type of the response that will be received from the OIDC provider.
Scopes	Preconfigured for most of the supported OIDC providers. Specify the access scope for the OIDC provider.

Field	Description
Client Authentication Method	Preconfigured for most of the supported OIDC providers. Select the authentication method to be used with the OIDC provider. Options are: <ul style="list-style-type: none"> <li>HTTP Basic authentication schema</li> <li>Data encoded form body</li> </ul>

#### Additional Support

Optional. Select the additional security parameters described in Table 7.

Table 7. Additional security options

Security parameter	Setting name for CLI	Description
NONCE	<code>nonce</code>	Enable the NONCE extension to prevent compromised requests from being used again (replayed).
Proof Key for Code Exchange (PKCE)	<code>pkce</code>	Enable the PKCE extension to allow public clients to mitigate the threat of having the authorization code intercepted.

#### Advanced Features

Optional. Select the advanced features described in Table 8.

Table 8. Advanced features

Feature / UI Label	Setting name for CLI	Description
Auto onboard	<code>auto_onboard</code>	Allow users to execute calls to APIs without logging in first, provided they present a valid token issued by the OIDC provider.
Always use the userinfo endpoint	<code>userinfo</code>	Configures the OIDC user registry to always fetch user data from the userinfo endpoint, if populated.
Use Portal as Endpoint for external OIDC provider traffic	<code>proxy_redirect</code>	When authenticating users, redirect the external OIDC provider to communicate with the Developer Portal instead of API Manager.
Return third-party access token	<code>proxy_access_token</code>	Include the third-party OIDC access token in the response. Note: Enable this setting for debugging purposes only. This setting is not recommended for use in a production environment. When this setting is enabled, the token size will increase when a request is made to API Connect using the token. The larger token size might exceed the HTTP protocol limit, resulting in an <code>ERR_HTTP2_PROTOCOL_ERROR</code> or <code>ERR_CONNECTION_CLOSED</code> error.
Return third-party id_token	<code>proxy_id_token</code>	Include the third-party OIDC id_token in the response. Note: Enable this setting for debugging purposes only. This setting is not recommended for use in a production environment. When this setting is enabled, the token size will increase when a request is made to API Connect using the token. The larger token size might exceed the HTTP protocol limit, resulting in an <code>ERR_HTTP2_PROTOCOL_ERROR</code> or <code>ERR_CONNECTION_CLOSED</code> error.
Token Relay	<code>token_relay</code>	Allow <code>access_token/refresh_token</code> to send in 302 redirect for logout
Userinfo POST	<code>post_userinfo</code>	If supported by your OIDC provider, by using HTTP POST method when contacting userinfo endpoint. Note: Not all OIDC providers support POST. Ensure that your OIDC provider supports this before you enable the feature.
Use IBM APIC token expiration setting from cloud	<code>override_provider_ttl</code>	Override the OIDC provider's token expiration setting with the access token and refresh token expiration settings that are configured in API Connect. For information on configuring the access token and refresh token expiration settings in API Connect, see <a href="#">Configuring timeouts for access tokens and refresh tokens</a> .
Disable hash verification (CLI only)	<code>disable_hash_verification</code>	Disable <code>at_hash, c_hash</code>

#### User Mapping

Fill in the settings as described in Table 9.

Note:

The User Mapping fields are preconfigured for most of the supported OIDC providers to minimize potential errors; use care when changing the settings. For the Standard OIDC option, contact your OIDC provider to obtain the details of the fields.

Table 9. User mapping settings


Field	Description
Username	The name of the field in the response token that contains the user's user name. Note: The username field must be unique for this OIDC registry, because it identifies the user in the system and cannot be changed.
Email	The name of the field in the response token that contains the user's email address.
First name	The name of the field in the response token that contains the user's given name.
Last name	The name of the field in the response token that contains the user's surname.

#### Redirect URI

The Redirect URI section lists the endpoints to which the OIDC authorization server will redirect the authorization code. There is one endpoint for each API Connect organization type: Cloud administration, Provider organization, and Consumer organization. The redirect endpoint is required when a user registers their application with the OIDC provider. For example, if this OIDC user registry is used by a provider organization, the user must register the provider organization's redirect endpoint with the OIDC provider. These read-only fields are provided for you to copy the endpoint values as required.

### Enabling the OIDC user registry

Complete the following steps to enable the new user registry for a specific catalog in the Developer Portal. Repeat this procedure for each catalog that will use the new registry.

- On the navigation pane, click  Manage.
- Select the catalog for which you will enable the new OIDC user registry.
- On the navigation pane, click the Catalog settings tab.

4. On the Settings page, click Onboarding.
5. On the Onboarding page, click Edit in the Catalog User Registries section.
6. On the Edit User Registry page, select the new OIDC user registry to enable for your catalog.
7. Click Save.

## Using the CLI to create an OIDC user registry

Use the developer toolkit CLI to create an organization-specific OIDC user registry when multi-factor authentication (MFA) is required.

Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access. You configure an OIDC user registry by first defining the registry details in a configuration file. You then use a CLI command to create the registry, passing the configuration file as parameter. To make the registry available to the Developer Portal, you must enable the registry in the associated Catalog.

Note:

- An OIDC registry, in common with a Local User Registry, cannot be used to secure APIs on the gateway.
- Because the interaction with the third party OIDC provider is handled by the Management server, the Management server is the application from the point of view of the third party OIDC provider. Your OIDC redirection endpoint, which is used by authorization server to send the token to the Management server, must be accessible to the OIDC provider through your firewall. When you register your application with the third party OIDC provider, you are required to supply the associated OIDC redirect URI, `https://consumer.mycompany.com/consumer-api/oauth2/redirect` for example. However, this information is not available until you have created your OIDC user registry in API Connect. You must therefore first register your application without this information, then update it later, as detailed in the instructions on this page.

### Logging in to the Management server

Before you can create an OIDC user registry, you must log in to your management server from the CLI. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details of the `apic login` command, see [Logging in to a management server](#).

### Defining your OIDC registry configuration

You define the configuration of your OIDC user registry in a YAML file. At a minimum, the YAML file must include the settings from the following list. For additional settings, see the tables provided with the UI version of the procedure in this topic.

```
title: registry_title
integration_url: oidc_integration_url
case_sensitive: case_sensitivity_setting
email_required: true_or_false
email_unique_if_exist: true_or_false
configuration:
 client_id: 'app_client_id'
 client_secret: 'my-client-secret'
 provider_type: oidc_provider_type
```

where:

- `registry_title` is your chosen descriptive title for the user registry.
- `oidc_integration_url` is the OIDC integration URL in your API Connect configuration. You can determine the OIDC integration URL by using the following CLI command:

```
apic integrations:list --server mgmt_endpoint_url --subcollection user-registry
```

- `case_sensitivity_setting` determines whether your user registry is case-sensitive. Valid values are:
  - `true`
  - `false`

To ensure proper handling of user name capitalization, you **must** ensure that your case-sensitivity setting here matches the setting on the backend OIDC provider:

- **Only** set `case_sensitive` to `true` if the backend OIDC provider supports case-sensitivity.
- Set `case_sensitive` to `false` if the backend OIDC provider does not support case-sensitivity.

Note: After at least one user has been onboarded into the registry, you cannot change this setting.

- `email_required` determines whether an email address is required as part of the user onboarding process. Valid values are:
  - `true`
  - `false`

If set to `true`, the source identity provider must supply the email address as part of the authentication process during onboarding.

Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.

- `email_unique_if_exist` determines whether email addresses must be unique within the user registry. Valid values are:
  - `true`
  - `false`

Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

- `app_client_id` is the client ID of the application that is registered with the OIDC server, and must be in string format.
- `my-client-secret` is the client secret of the application that is registered with the OIDC server, and must be in string format.
- `oidc_provider_type` is the type of OIDC provider; specify one of the following values:
  - `facebook`
  - `github`
  - `google`
  - `linkedin`
  - `slack`
  - `twitter`
  - `windows_live`
  - `standard`

Use the `standard` provider type for any OIDC provider that is compliant with the OIDC standard.

Note: If the provider type is `standard`, you must include the following additional properties in the `configuration` section of your YAML file:

```
authorization_endpoint: 'oidc_auth_endpoint'
token_endpoint:
 endpoint: 'oidc_token_endpoint'
```

where:

- `oidc_auth_endpoint` is the authorization endpoint on the OIDC server, and must be in string format.
- `oidc_token_endpoint` is the token endpoint on the OIDC server, and must be in string format.

## Default OIDC configurations

For each OIDC provider type, API Connect assumes a default configuration, but you can override the default configuration properties in your YAML file. The default configurations are as follows:

- Facebook

```
authorization_endpoint: 'https://www.facebook.com/v3.1/dialog/oauth'
token_endpoint:
 endpoint: 'https://graph.facebook.com/v3.1/oauth/access_token'
userinfo_endpoint:
 endpoint: 'https://graph.facebook.com/me'
scope: email public_profile
field_mapping:
 username: email
 email: email
 last_name: last_name
 first_name: first_name
```

- Github

```
authorization_endpoint: 'https://github.com/login/oauth/authorize'
token_endpoint:
 endpoint: 'https://github.com/login/oauth/access_token'
userinfo_endpoint:
 endpoint: 'https://api.github.com/user'
scope: 'read:user user:email'
field_mapping:
 username: login
 email: email
 last_name: name
 first_name: name
```

- Google

```
authorization_endpoint: 'https://accounts.google.com/o/oauth2/v2/auth'
token_endpoint:
 endpoint: 'https://www.googleapis.com/oauth2/v4/token'
scope: openid profile email
field_mapping:
 username: email
 email: email
 last_name: family_name
 first_name: given_name
```

- LinkedIn

```
authorization_endpoint: 'https://www.linkedin.com/oauth/v2/authorization'
token_endpoint:
 endpoint: 'https://www.linkedin.com/oauth/v2/accessToken'
userinfo_endpoint:
 endpoint: 'https://api.linkedin.com/v1/people/~:(id,first-name,last-name,picture-url,public-profile-url,email-address)?format=json'
scope: r_basicprofile r_emailaddress
field_mapping:
 username: emailAddress
 email: emailAddress
 last_name: lastName
 first_name: firstName
credential_location: form_body
```

- Slack

```

authorization_endpoint: 'https://slack.com/oauth/authorize'
token_endpoint:
 endpoint: 'https://slack.com/api/oauth.access'
userinfo_endpoint:
 endpoint: 'https://slack.com/api/users.identity'
scope: identity.basic identity.email
field_mapping:
 username: user.email
 email: user.email
 last_name: user.name
 first_name: user.name

```

- Twitter

```

request_endpoint: https://api.twitter.com/oauth/request_token'
authorization_endpoint: https://api.twitter.com/oauth/authenticate'
token_endpoint:
 endpoint: 'https://api.twitter.com/oauth/access_token'
userinfo_endpoint:
 endpoint: 'https://api.twitter.com/1.1/account/verify_credentials.json'
oauth_signature_method: 'HMAC-SHA1'
field_mapping:
 email: email
 first_name: name
 last_name: name
 username: screen_name

```

- Windows Live

```

authorization_endpoint: 'https://login.microsoftonline.com/common/oauth2/v2.0/authorize'
token_endpoint:
 endpoint: 'https://login.microsoftonline.com/common/oauth2/v2.0/token'
scope: openid offline_access profile email
field_mapping:
 email: email
 username: preferred_username
 first_name: name
 last_name: name

```

- Standard

```

response_type: code
scope: openid
field_mapping:
 username: sub
 email: email
 last_name: family_name
 first_name: given_name
credential_location: auth_header

```

Although this is the default configuration for the `standard` provider type, you should contact your OIDC provider to obtain the details of the fields that you need to define.

## Creating your OIDC user registry

To create your OIDC user registry, use the following CLI command:

```
apic user-registries:create --server mgmt_endpoint_url --org organization_name oidc_config_file
```

where:

- `mgmt_endpoint_url` is the platform API endpoint URL.
- `organization_name` is the value of the `name` property of your provider organization.
- `oidc_config_file` is the name of the YAML file that defines the configuration of your OIDC user registry.

On completion of the registry creation, the command displays the following summary details:

```
registry_name registry_url
```

By default, the `registry_name` is derived from the `title` property in the configuration YAML file but you can override this by including a `name` property in the file. The `registry_url` is an internal URL that API Connect assigns to the registry.

After you have created your OIDC user registry, you must update your application registration with the third party OIDC provider to include the OIDC redirect URI; you can obtain this information by using the following command, which displays the details of the registry in the command window:

```
apic user-registries:get --server mgmt_endpoint_url --org organization_name registry_name --output -
```

The required `oidc_redirect_uri` value is in the `consumer:` section; for example:

```
consumer:
 oidc_redirect_uri: https://consumer.mycompany.com/consumer-api/oauth2/redirect
```

## Enabling your OIDC registry in a Catalog

To make your OIDC registry available for onboarding and authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. Complete the following steps:

1. Determine the URL of your OIDC user registry by using the following command:

```
apic user-registries:list --server mgmt_endpoint_url --org organization_name
```

2. Enter the following command (the terminating hyphen character means that the command takes input from the command line):

```
apic configured-catalog-user-registries:create --server mgmt_endpoint_url --org organization_name --catalog catalog_name -
```



where `catalog_name` is the value of the `name` property of the required Catalog. The command returns

```
Reading CONFIGURED_CATALOG_USER_REGISTRY_FILE arg from stdin
```

3. Enter the following data, followed by a new line:

```
user_registry_url: oidc_registry_url
```

where `oidc_registry_url` is the URL of your OIDC registry, obtained in step 1.

4. Press **CTRL D** to terminate the input.

For details of all the `apic user-registries` and `apic configured-catalog-user-registries` commands, see [apic user-registries](#) and [apic configured-catalog-user-registries](#).

You can also complete the operations described in this topic by using the API Connect REST APIs; see the [API Connect REST API documentation](#).

---

## Modifying the configuration details for a user registry

You modify the configuration details for a user registry by using the User Registries page in the API Manager user interface of IBM® API Connect.

### About this task



If you want to authenticate an API Connect Catalog with a user registry that is not yet defined in API Connect, you define the registry when you configure the Catalog; for details, see [Working with user registries](#).

Note: User registries shared from the Cloud Manager cannot be edited.

To modify the configuration details for a user registry that is already defined, you use the User Registries page.

### Procedure

To modify the configuration details for a user registry, complete the following steps:

1. In the API Manager, click  Resources, and then select User Registries.
2. On the User Registries page, click the options icon  for the user registry that you want to edit, then click Edit. The details of the registry are displayed.
3. Modify the configuration details as required, then click Save to save your changes.

---

## Password lockout criteria

You can be locked out of your account if you attempt to log in and fail consecutively.

Note: Account lock out applies only for local user registries.

The length of time that you are locked out of using the account is based on the number of consecutive attempts that you fail. The length of time increases as the number of consecutive failed attempts increases.

For example, you can be locked out for 15 seconds if you have five consecutive failed attempts, or 32 minutes for 12 consecutive failed attempts.

Note: External user registries, such as LDAP, might enforce their own lockout criteria.


---

## Configuring a native OAuth provider

Native OAuth providers are configured and managed by you within your cloud.

### About this task

A native OAuth provider object provides settings for OAuth processing operations such as generating and validating OAuth tokens. An OAuth provider object is referenced by an OAuth security definition to protect an API. When a native OAuth provider is used, the OAuth operations are performed natively by API Connect.

Every OAuth provider object has a backing API. Your configuration here automatically updates the OpenAPI document of the API. You can edit the OpenAPI document directly by navigating to the  Resources > OAuth Providers page, selecting your OAuth provider, then clicking API Editor.

Note: Take care when modifying the code directly on the Source tab of the API Editor because validation is limited. For example:

- If you change the name of auto generated assembly actions in the source code, the assembly will be prevented from updating dynamically when the OAuth provider settings are modified.
- You must ensure that the OAuth provider name matches the value specified in the `oauth-provider-settings-ref` field in each OAuth assembly action.

When a published API references an OAuth provider object, the backing API is automatically made available in the gateway.

Token requests and `client_id` checks:

If you are calling an OAuth endpoint, the sequence of `client_id` checks in the token request is as follows:

1. Check both body and query.
  - If found in only the body, validate and return a 200 or appropriate return code.

- If found in only the query, return a **wrong location** error.
  - If found in both the body and the query, return a **more than one location** error.
- When not found in the body or the query, check the **Authorization** header.
    - If found in this header, validate and return a **200** or appropriate return code.
  - Not found, return the appropriate error and code.


The best location for the `client_id` is in the request body.  
 One of the following roles is required to configure a native OAuth provider:

- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

Note:


- The OAuth provider logs Analytics data for failure cases, but does not log successful cases. Activity log policies that call for logging of Analytics data upon success do not apply for the OAuth provider.
- You must ensure the OAuth Provider is configured in the Sandbox Catalog before using the OAuth Provider in a non-Sandbox Catalog.

## Procedure

- In the API Manager, click  Resources.
- Click OAuth Providers > Add > Native OAuth provider.
  - Complete the following parameters for the first screen, then click Next.

Field	Description
Title	Enter a title for the native OAuth provider.
Name	This field is auto-populated by the system.
Description (optional)	Enter a brief description.
Base path (optional)	The base path is the URL segment of the API that is shared by all operations in the API. It does not include the host name or any additional segments for paths or operations. The base path must be unique for a given catalog. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
Gateway Type	Select the gateway type, either DataPower® Gateway (v5 compatible) or DataPower API Gateway. For information about types of gateways, see <a href="#">API Connect gateway types</a> .  OAuth Providers apply to one gateway type.



- In the next screen, enter the following additional configuration parameters, then click Next.

Field	Description
Authorize Path	<code>/oauth2/authorize/</code> is the standard OAuth endpoint to login to account
Token Path	<code>/oauth2/token/</code> is the standard OAuth endpoint to exchange code for access token.
Supported grant types	<ul style="list-style-type: none"> <li>• Implicit - An access token is returned immediately without an extra authorization code exchange step.</li> <li>• Application - Application to application. Corresponds to the OAuth grant type "Client Credentials." Does not require User Security.</li> <li>• Access code - An authorization code is extracted from a URL and exchanged for an access code. Corresponds to the OAuth grant type "Authorization Code."</li> <li>• Resource owner - Password - The user's username and password are exchanged directly for an access token, so can only be used by first-party clients.</li> <li>•  Resource owner - JWT - A verified signed JSON Web Token is exchanged directly for an access token.</li> </ul>
Supported client types	<ul style="list-style-type: none"> <li>• Confidential - Client can maintain secure credentials on a secure server</li> <li>• Public - Client credentials are not secure.</li> </ul>

- Enter the scopes in the next screen. A scope becomes an option in the request and response for an access token. Click Add to add additional fields for scopes. Click Next when done.

Field	Description
sample_scope_1	Scope for token
additional scopes	Scope for token

- Enter the parameters for User Security in the next screen. Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization. User Security is not required for the Application grant type. Click Next when done.

Field	Description
Identity Extraction	Determines how the user credential is extracted: <ul style="list-style-type: none"> <li>• Basic Authentication - HTTP basic authentication (requires no additional configuration)</li> <li>• Default HTML Form - Use default login form for user name and password</li> <li>•  Context variable - Specify which variable contains the user name and password. API Connect OAuth context variables as listed here <a href="#">API Connect context variables</a> Note: DataPower API Gateway only. Not available for DataPower Gateway (v5 compatible).</li> <li>• Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form. For instructions on creating a custom form, see <a href="#">Creating a custom HTML login form for user security</a>.</li> <li>• Redirect - Enter an endpoint to redirect to a third-party identity provider. For more information, see <a href="#">Authenticating and authorizing through a redirect URL</a>.</li> <li>•  Disabled - do not collect the user credential</li> </ul> Note: If you use either the Default HTML Form or Redirect identity extraction methods, the response from the redirect endpoint <b>must</b> maintain the order of the query parameters before the <code>state_nonce</code> query parameter, otherwise the authorization fails.
Authentication	Authenticate application users with a user registry. Select an LDAP or Authentication URL user registry or create the SampleAuthURL User Registry. For a DataPower API Gateway, you have the option to disable authentication with a user registry.

Field	Description
Authorization	<p>The following methods for extracting the user credential are available:</p> <ul style="list-style-type: none"> <li>Authenticated - Authorize authenticated users automatically.</li> <li>Default HTML Form - Use default HTML form to authorize.</li> <li>Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form. For instructions on creating a custom form, see <a href="#">Creating a custom HTML authorization form for user security</a>.</li> <li><span style="border: 1px solid green; padding: 2px;">API Gateway only</span> Disabled - Disable authorization.</li> </ul>

- Review the Summary for the native OAuth provider configuration.
- Click Back to make changes.
- Click Finish to save the basic configurations and to proceed to the Advanced Parameters for a native OAuth Provider.

## Results

You can use the OAuth Provider to secure the APIs in catalog.

- [Configuring basic settings for a native OAuth provider](#)  
You can update the identification details and basic configuration settings for a native OAuth provider.
- [Configuring scopes for a native OAuth provider](#)  
Access tokens contain authorization for specific scopes.
- [Configuring user security for a native OAuth provider](#)  
Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization.
- [Configuring tokens for a native OAuth provider](#)  
Set time to live for access tokens and refresh tokens, and a time period for maximum consent for all tokens.
- [Configuring token management and revocation for a native OAuth provider](#)  
Select whether to use a native gateway (DataPower) or third party endpoint for token revocation.
- [Configuring introspection for a native OAuth provider](#)  
Define an introspection path to allow the metadata for an access token to be examined.
- [Configuring metadata for a native OAuth provider](#)  
Use Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.
- [Configuring the OIDC parameters for a native OAuth provider](#)  
Open ID Connect (OIDC) provides an additional authentication protocol based on OAuth 2.0. OIDC provides user information encoded in a JSON Web Token, or JWT.
- [Editing a native OAuth provider by using the API Editor](#)  
You can edit the source and assembly policies for the Native OAuth Provider using the API editor.

## Configuring basic settings for a native OAuth provider


You can update the identification details and basic configuration settings for a native OAuth provider.

### About this task

One of the following roles is required to configure the basic settings for a native OAuth Provider:

- Organization Administrator
- Owner
- Custom role with the Settings\_>\_Manage permissions

You can select the basis settings pages for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the basic settings for an existing native OAuth provider. If you want to update the basic settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

- Click  Resources > OAuth Providers.
- Select the required native OAuth provider.

### Procedure

- To modify the identification details, click Info in the sidebar menu, then update the following fields as required:

Field	Description
Title	Enter a title for the native OAuth provider.
Name	This field is auto-populated by the system.
Description (optional)	Enter a brief description.
Base path (optional)	The base path is the URL segment of the API that is shared by all operations in the API. It does not include the host name or any additional segments for paths or operations. The base path must be unique for a given catalog. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.

- To modify the basic configuration settings, click Configuration in the sidebar menu, then update the following fields as required:

Field	Description
Authorize Path	/oauth2/authorize/ is the standard OAuth endpoint to login to account
Token Path	/oauth2/token/ is the standard OAuth endpoint to exchange code for access token.

Field	Description
Supported grant types	<ul style="list-style-type: none"> <li>Implicit - An access token is returned immediately without an extra authorization code exchange step.</li> <li>Application - Application to application. Corresponds to the OAuth grant type "Client Credentials." Does not require User Security.</li> <li>Access code - An authorization code is extracted from a URL and exchanged for an access code. Corresponds to the OAuth grant type "Authorization Code."</li> <li>Resource owner - Password - The user's username and password are exchanged directly for an access token, so can only be used by first-party clients.</li> <li><span style="border: 1px solid green; padding: 2px;">API Gateway only</span> Resource owner - JWT - A verified signed JSON Web Token is exchanged directly for an access token. Tip: Selecting only the Resource owner - JWT grant type when defining a native OAuth provider in the user interface is not supported and results in an invalid configuration. To avoid this problem, additionally select either the Access code or the Resource owner - Password grant type.</li> </ul> <p>Note: If you plan to configure OpenID Connect (OIDC) for a native OAuth provider, you must include at least one of the following grant types: Implicit, Access code.</p>
Supported client types	<ul style="list-style-type: none"> <li>Confidential - Client can maintain secure credentials on a secure server</li> <li>Public - Client credentials are not secure.</li> </ul>

DataPower Gateway (v5 compatible) only Note: If the gateway type is DataPower® Gateway (v5 compatible) and, when the native OAuth provider was created, only the Application grant type was selected, you cannot add further grant types until you configure the user security settings. In particular, you must specify the user registry for authenticating application users. To configure the user security settings, complete the following steps:

- a. Click User Security in the sidebar menu, then click Edit.
- b. Update the user security settings as required; for more details, see [Configuring user security for a native OAuth provider](#).
- c. Click Save when done.

3. Click Save when done.

## Configuring scopes for a native OAuth provider

Access tokens contain authorization for specific scopes.

### About this task

The client applications can request only the scopes or a subset of the scopes that you define here. The scopes are included in the access tokens that are generated from the provider. When an OAuth protected API is invoked, the gateway checks the scopes carried in the access tokens against the list of allowed scopes in the security definition for the API to determine whether to grant access.


In addition, you can enforce advanced scope checks. The advanced scope check URLs are invoked after application authentication or after user authentication based on which URLs are configured. The final scope permission that is granted by the access token is the result of all scope checks.

Per IETF RFC 6749, the value of the scope parameter is a list of space-delimited, case-sensitive strings. For more information, see [The OAuth 2.0 Authorization Framework](#).

One of the following roles is required to configure scopes for a native OAuth Provider:

- Organization Administrator
- Owner
- Custom role with the Settings\_>\_Manage permissions

You can select the scope settings page for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the scope settings for an existing native OAuth provider. If you want to update the scope settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

- Click  Resources\_>\_OAuth Providers.
- Select the required native OAuth provider.

### Procedure

- Click Scopes in the sidebar menu. The currently configured scopes are listed. Review and update the scopes as required.

Field	Description
sample_scope_1	Scope for token
additional scopes	Scope for token

In the Default scopes section, select the default scopes to be used if the API request doesn't contain any scopes. If no default scope is defined, and the request doesn't contain a scope, an invalid scope error is returned for the request.

If the user authorization method is set to Default HTML Form in the User Security settings, all scopes specified here are added automatically to the authorization consent form.

- Advanced scope check before token generation. This setting specifies the scope check endpoint where additional scope verification is performed in addition to the basic scopes. The advanced scope check URLs are invoked after application authentication or after owner authentication based on which URLs are configured. The scopes are included in the token and will overwrite any previous scopes.

Field	Description
Application scope check	Allow extra verification by running a scope check from an endpoint. Enter the endpoint and an optional TLS Profile to use for an application scope check.
Owner scope check	Further refine the scope with an additional check. Enter the endpoint and an optional TLS Profile to use for an owner scope check.

For more information about scope, see [Scope](#)

- Advanced scope check after token generation. This setting specifies an additional scope check at the API consumer level to verify compliance with the scope requirements of the API.

Field	Description

Field	Description
Enabled	Select the check box to enable the advanced scope check after token validation. Enter an optional default validator endpoint.
Use endpoint from API	Select the check box to use the endpoint from the API, or clear the check box to override the endpoint from the API.

For more information about scope, see [Scope](#)

4. Click Save when done.

## Results

You can use the OAuth Provider with these scopes to secure the APIs in catalog.

## Configuring user security for a native OAuth provider

Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization.


### About this task

User security authenticates the user. It is required for the Implicit, Access code, and Resource owner - Password grant types. It is not used for the Application or Resource owner - JWT grant types.

One of the following roles is required to configure user security for a native OAuth Provider:




- Organization Administrator
- Owner
- Custom role with the Settings\_>Manage permissions

You can select the user security settings page for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the user security settings for an existing native OAuth provider. If you want to update the user security settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click  Resources > OAuth Providers.
2. Select the required native OAuth provider.

### Procedure

1. Click User Security in the sidebar menu.
2. Specify the following parameters for User Security. Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization. User Security is not required for the Application or Resource owner - JWT grant types. Click Next when done.

Field	Description
Identity Extraction	<p>Determines how the user credential is extracted:</p> <ul style="list-style-type: none"> <li>• Basic Authentication - HTTP basic authentication (requires no additional configuration)</li> <li>• Default HTML Form - Use default login form for user name and password</li> <li>•  Context variable - Specify which variable contains the user name and password. API Connect OAuth context variables as listed here <a href="#">API Connect context variables</a></li> <li>• Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form. For instructions on creating a custom form, see <a href="#">Creating a custom HTML login form for user security</a>.</li> <li>• Redirect - Enter an endpoint to redirect to a third-party identity provider. For more information, see <a href="#">Authenticating and authorizing through a redirect URL</a>.</li> <li>•  Disabled - do not collect the user credential</li> </ul> <p>Note: If you use either the Default HTML Form or Redirect identity extraction methods, the response from the redirect endpoint <b>must</b> maintain the order of the query parameters before the <code>state_nonce</code> query parameter, otherwise the authorization fails.</p>
Authentication	<p>Authenticate application users with a user registry. Select an LDAP or Authentication URL user registry or create the SampleAuthURL User Registry.</p>
Authorization	<p>Various methods may be used to authorize application users. For a DataPower® API Gateway, the following methods for extracting the user credential are available:</p> <ul style="list-style-type: none"> <li>• Authenticated - Authorize authenticated users automatically.</li> <li>• Default HTML Form - Use default HTML form to authorize. If you select the Default HTML Form method, all scopes that are specified in the Scopes settings are added automatically to the authorization consent form.</li> <li>• Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form.</li> <li>•  Disabled - Disable authorization.</li> </ul>

3. Click Save when done.

## Results

You can use the OAuth Provider to secure the APIs in catalog.

## Configuring tokens for a native OAuth provider

Set time to live for access tokens and refresh tokens, and a time period for maximum consent for all tokens.


## About this task

Access tokens are granted to the client application to allow the application to access resources on behalf of the application user. Refresh tokens are issued to the client to obtain a new access token when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or more narrow scope. You can also specify how long the consent given by the combination of any number of access and refresh token remains valid.

One of the following roles is required to configure tokens for a native OAuth Provider:




- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

You can select the token settings page for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the token settings for an existing native OAuth provider. If you want to update the token settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click  Resources > OAuth Providers.
2. Select the required native OAuth provider.

## Procedure

1. Click Tokens in the sidebar menu.
2. Define the settings to configure tokens.

Field	Description
Access tokens time to live	Enter the expiration time period in seconds for access tokens.
 One time use access token	Click the check box to enable one time use for the access token. Access tokens are multiple use by default which allows them to be used for multiple requests. When one time use is enabled, the access token will be consumed after one use. The OAuth flow will need to be repeated to obtain another access token. Note: If you select this option, you must also enable token management; see one of the following topics, depending on the user interface you are using: <ul style="list-style-type: none"> <li>• <a href="#">Configuring token management and revocation for a native OAuth provider</a> (Cloud Manager)</li> <li>• <a href="#">Configuring token management and revocation for a native OAuth provider</a> (API Manager)</li> </ul>
Refresh tokens	Click the check box to enable Refresh tokens. Set the Count to limit the number of times a refresh token can be issued. Set the Refresh Token Time to Live value to determine the time to live, or expiration time period, for each refresh token in seconds.
One time use refresh token	Clear the check box to disable one time use for the refresh tokens. Refresh tokens are one time use by default which allows them to be used one time only to generate an access token and a new refresh token. When refresh token one time use is disabled then the refresh token count is limited to one and the refresh token can be used multiple times to generate new access tokens, however, another refresh token will not be generated unless the initial OAuth flow (Authorization Code or Password) is repeated. Note: If you select this option, you must also enable token management; see one of the following topics, depending on the user interface you are using: <ul style="list-style-type: none"> <li>• <a href="#">Configuring token management and revocation for a native OAuth provider</a> (Cloud Manager)</li> <li>• <a href="#">Configuring token management and revocation for a native OAuth provider</a> (API Manager)</li> </ul>
Maximum consent	Click the check box to enable Maximum consent and enter the Maximum Consent Time to Live value in seconds. This is the time to live, or expiration time period, for all tokens, both access and refresh.
 Token secret	Click the check box to select the Shared Secret which was configured for the gateway. If no Shared Secret was entered in the Gateway Configuration, then enter an key name and key value to use as the token secret.
 Proof Key for Code Exchange	Proof Key for Code Exchange (PKCE) is a method to protect OAuth 2.0 public clients from an authorization code interception attack when they use Authorization Code grant requests. You can enable this extension when deploying with the DataPower® API Gateway. For more information, see <a href="#">RFC 7636</a> .  Select the options for your OAuth Providers: <ul style="list-style-type: none"> <li>• Enable proof key for code exchange If selected, enforces PKCE when submitted in Authorization Code grant requests.</li> <li>• Always required If selected, requires PKCE in all Authorization Code grant requests.</li> <li>• Allow plain Select this check box to allow the plain challenge method in Authorization Code grant requests.</li> </ul>

3. Click Save when done.

## Results

You can use the OAuth Provider to secure the APIs in a catalog.

## Configuring token management and revocation for a native OAuth provider

Select whether to use a native gateway (DataPower) or third party endpoint for token revocation.

## About this task


Token management enables you to prevent replay attacks by configuring token revocation. API Connect supports token revocation using a native gateway (DataPower) or a third party endpoint. For a native gateway, quota enforcement is used to manage tokens. For a third party endpoint, a URL to an external service is used to manage tokens.

For more information, see the IETF RFC 7009 [OAuth 2.0 Token Revocation](#).

One of the following roles is required to configure token management and revocation for a native OAuth Provider:

- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

You can select the token management settings page for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the token management settings for an existing native OAuth provider. If you want to update the token management settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click  Resources > OAuth Providers.
2. Select the required native OAuth provider.

## Procedure

---

1. Select Token Management in the sidebar menu.
2. Enable token management by selecting the check box.
3. From the Type list, select either Native or External. Native points to DataPower as the token storage location; External points to a revocation URL for token storage.  
Note: If you are using the API Manager user interface then, for the External option to be available, you must be using DataPower® API Gateway Version 10.0.1.0 or later, and the gateway service must be enabled in the Sandbox Catalog; for details on how to enable a gateway service in a Catalog, see [Creating and configuring Catalogs](#)
4. For Native, select one or both of the Resource owner revocation path and Client revocation path.
  - Resource owner revocation path - Uses the standard OAuth revocation path to allow the resource owner (end user) to revoke the application permission.
  - Client revocation path - Uses the standard OAuth revocation path to allow the client (application) to revoke a single token when the application closes.For more information about managing tokens with the Native DataPower Gateway, see [Token management with the native DataPower Gateway](#).
5. For External, the settings depend on the gateway type, as follows:  
DataPower API Gateway:
  - Endpoint - the URL of the external management endpoint.
  - TLS Client Profile (optional) - the TLS client profile to secure connections.
  - Security - how to secure connections. The only supported method is basic authentication.
  - Basic authentication username (optional) - the user name for authentication.
  - Basic authentication password (optional) - the password for authentication.
  - Basic authentication request header name (optional) - the request header that contains the authentication string; if you supply both a request header name and user name/password, the request header authentication method is used.
  - Custom header pattern (optional) - the name pattern of the headers to use for sending additional information to the external management service.
  - Cache type - the cache type to control whether and how to cache positive responses. If you select Time to live, specify how long to keep responses in the cache; the default value is 900 seconds.
  - Fail on error - if selected, processing is stopped if the connection to the external management service fails.Note: For details of the JSON format that is required when exchanging messages with the external management service, see [JSON format to exchange messages with the external management service](#).  
DataPower Gateway (v5 compatible):
  - Endpoint - Enter the URL to an external web server that contains information about access or refresh tokens. API Connect calls the URL to determine if the associated token can be trusted. The token server then checks a token *blocklist* (a data store of inactive tokens) to ensure that the token is still valid. If the token is still valid, API Connect continues the processing. For more information see [Token revocation](#).
  - TLS Client Profile - Select a TLS profile to verify the external endpoint.
6. Click Save when done.

## Results

---

You can use the OAuth Provider to secure the APIs in a catalog.

## Configuring introspection for a native OAuth provider

Define an introspection path to allow the metadata for an access token to be examined.

### About this task

---


Token introspection allows an authorized holder of an access token to examine the contents of tokens using an introspection path. The access token to introspect must be obtained through the native OAuth provider. Introspection provides context for the token by allowing an authorized protected resource to query the authorization server to determine the set of metadata for a given token. The metadata includes whether or not the token is currently active, the scopes assigned to the token, and the authorization context in which the token was granted (including who authorized the token and which client it was issued to). API Connect token introspection conforms to IETF RFC 7662. See [OAuth 2.0 Token Introspection](#).

One of the following roles is required to enable introspection for a native OAuth Provider:

- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

You can select the introspection settings page for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the introspection settings for an existing native OAuth provider. If you want to update the introspection settings for an existing native OAuth

provider, complete the following steps before following the procedure described in this topic:

1. Click  Resources > OAuth Providers.
2. Select the required native OAuth provider.

## Procedure

---

1. Click Introspection in the sidebar menu.
2. Select the check box to enable Introspection. The OAuth standard path for introspection, /oauth2/introspect is automatically entered. This path will be used when another entity inspects the token contents.
3. Click Save when done.

## Results

---

Tokens will be queried using the /oauth2/introspect path. You can use the OAuth Provider to secure the APIs in a catalog.

## Configuring metadata for a native OAuth provider

---

Use Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.

## About this task

---

Configure an Authentication URL or an External URL from which custom metadata is collected for inclusion in the token. The metadata is either stored inside the access token or it is sent along with the access token to the client application. For more information about how the metadata is collected, see [OAuth external URL and authentication URL](#).


Following are examples of metadata that can be included with the access token:

- Metadata about the authenticated resource owner
- Grant type that was used to obtain the token
- A confirmation code to be provided to the client application

One of the following roles is required to configure metadata collection for a native OAuth Provider:



- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

You can select the metadata settings page for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the metadata settings for an existing native OAuth provider. If you want to update the metadata settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click  Resources > OAuth Providers.
2. Select the required native OAuth provider.

## Procedure

---

1. Click Metadata in the sidebar menu.
2. Select Collect metadata to enable metadata collection.
3. The Authentication URL user registry is selected by default and is required. For more information about the Authentication URL, see [Authentication URL user registry](#).
4. Select the External URL to collect metadata from an external URL. Enter the endpoint and an option TLS Client Profile.
5.  If required, override the default Header name token value. The value of this header, if returned in the response from the OAuth endpoint, is placed in the response payload and indicated as **metadata**.
6.  If required, override the default Header name payload value. The value of this header, if returned in the response from the OAuth endpoint, is placed within the access token and indicated as **miscinfo**.
7. Save the OAuth Provider.
8. Click Save when done.

## Results

---

You can use the OAuth Provider to secure the APIs in a catalog.

## Configuring the OIDC parameters for a native OAuth provider

---

Open ID Connect (OIDC) provides an additional authentication protocol based on OAuth 2.0. OIDC provides user information encoded in a JSON Web Token, or JWT.

## About this task

---




When you enable OpenID connect, a template is provided for generating ID tokens along with access tokens and the required assembly policies are automatically created. You can customize the policies to suit your needs in the API Editor. The sample key provided is for test purposes only and is used to sign the JWT token.

One of the following roles is required to configure an OIDC template for a native OAuth Provider:

- Organization Administrator
- Owner
- Custom role with the System\_>\_Manage permissions

Note: You can configure OIDC parameters only if the selected grant types for the native OAuth provider include at least one of the Implicit or Access code grant types; see [Configuring basic settings for a native OAuth provider](#).

You can select the OIDC settings page for a native OAuth provider immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the OIDC settings for an existing native OAuth provider. If you want to update the OIDC settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click  Resources\_>\_OAuth Providers.
2. Select the required native OAuth provider.

## Procedure

1. Click OpenID Connect in the sidebar menu.
2. Select the initial check box to configure an OIDC Template. Enter the following parameters:

Field	Description
<input checked="" type="checkbox"/> API Gateway only Support hybrid response types (optional)	Select the response types for the OpenID Connect hybrid flow to be supported by this OAuth provider.
<input checked="" type="checkbox"/> API Gateway only Auto Generate OIDC API Assembly	Select this option to generate the full OIDC assembly. Leave this option unselected to simply enable OIDC support in the OAuth provider and allow the developer to implement their own assembly.
<input type="text"/> DataPower Gateway (v5 compatible) only ID token issuer	Descriptive text to indicate the source of the key.
<input type="text"/> DataPower Gateway (v5 compatible) only ID token signing key	Specify the JSON Web Key (JWK) to be used to sign the ID token.
<input type="text"/> DataPower Gateway (v5 compatible) only ID token signing algorithm	Select the algorithm used to sign the token.

3. Click Save when done. You can edit the policies by using the API Editor.

## Results

You can use the OAuth Provider to secure the APIs in a catalog.

## Editing a native OAuth provider by using the API Editor

You can edit the source and assembly policies for the Native OAuth Provider using the API editor.

## About this task

If you have configured an OIDC template, you can customize it in API Editor. In the API Editor, the Source tab allows you to edit the code for the configuration using a text editor. The API Assemble tab provides a graphical drag-and-drop editor (identical to the one in API Manager) that allows you to add additional elements to the assembly for the OAuth Provider.


Note: Take care when modifying the code directly on the Source tab of the API Editor because validation is limited. For example:

- If you change the name of auto generated assembly actions in the source code, the assembly will be prevented from updating dynamically when the OAuth provider settings are modified.
- You must ensure that the OAuth provider name matches the value specified in the `oauth-provider-settings-ref` field in each OAuth assembly action.

One of the following roles is required to configure tokens for a native OAuth Provider:

- Organization Administrator
- Owner
- Custom role with the Settings\_>\_Manage permissions

You can modify the native OAuth provider configuration by selecting the API Editor page immediately on completion of the creation operation detailed in [Configuring a native OAuth provider](#), or you can update the configuration for an existing native OAuth provider. If you want to update the configuration for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click  Resources\_>\_OAuth Providers.
2. Select the required native OAuth provider.

## Procedure

1. Click API Editor in the sidebar menu.
2. In the Source tab, view and edit the policies to customize the behavior for the OAuth provider.

3. In the API Assemble tab, use the drag and drop editor to add additional policies to the OIDC behavior.

Note: If you add a policy that references a TLS profile, an **invoke** policy for example, then when you publish an API that uses this OAuth provider, you must ensure that the TLS profile is enabled for the Catalog to which you publish the API. For details on how to enable a TLS profile in a Catalog, see [Creating and configuring Catalogs](#).

4. Save the edits.
5. Click Save when done.

## Results

You can use the OAuth Provider to secure the APIs in a catalog.

## Configuring a third-party OAuth provider


Enter the secure endpoints to provide OAuth authentication from a third party.

### About this task

One of the following roles is required to configure OAuth Providers:

- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

### Procedure

1. In the API Manager, click  Resources.
2. Select OAuth Providers > Add > Third party OAuth Provider.
  - a. Complete the following parameters for the first screen and click Next.

Field	Description
Title	Enter a descriptive title for the gateway service. This title will be displayed on the screen.
Name	This field is auto-populated by the system and used as the internal field name.
Supported grant types	Select from the following options: <ul style="list-style-type: none"> <li>• Implicit: An access token is returned immediately without an extra authorization code exchange step.</li> <li>• Application: Application to application. Corresponds to the OAuth grant type Client Credentials. Does not require User Security.</li> <li>• Access code: An authorization code is extracted from a URL and exchanged for an access code. Corresponds to the OAuth grant type Authorization Code.</li> <li>• Resource owner - Password: The user's username and password are exchanged directly for an access token, so can only be used by first-party clients.</li> <li>• <span style="border: 1px solid green; padding: 2px;">API Gateway only</span> Resource owner - JWT: a JSON Web Token (JWT) Bearer Token is used as a means for requesting an OAuth 2.0 access token, and for client authentication, as defined by the <a href="#">JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants</a>.</li> </ul>
Gateway type	Select the gateway type, either DataPower® Gateway (v5 compatible) or DataPower API Gateway. For information about types of gateways, see <a href="#">API Connect gateway types</a> .  OAuth Providers apply to one gateway type.

- b. Specify configuration settings for the endpoints.

Field	Description
Authorization URL	An authorization URL where the resource owner grants authorization to the client application to access a protected resource. Example:  <code>https://example.com/oauth2/authorize</code>
Token URL	A token request URL where the client application exchanges an authorization grant for an access token. Example:  <code>https://example.com/oauth2/token</code>
Introspect URL	The introspection URL is where the API gateway validates the access tokens that are issued by the third party provider. Example:  <code>https://example.com/oauth2/introspect</code>  For more information on integrating third party OAuth providers for introspection, see <a href="#">OAuth introspection for third-party OAuth providers</a> .
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> Introspect cache type	The cache type determines how long responses from the third party provider are cached, if at all. Select one of the following options: <ul style="list-style-type: none"> <li>• No cache (default): Responses are not cached.</li> <li>• Protocol: Defined by the <code>cache-control</code> header in the provider response.</li> <li>• Time to live: Defined by the provider.</li> </ul>
<span style="border: 1px solid green; padding: 2px;">API Gateway only</span> Cache Time to Live	The length of time, in seconds, for which provider responses are cached, if the Introspect cache type is set to Time to live. The default value is 900.
TLS Profile (optional)	Select an optional TLS profile for communicating with the third party provider.
Security	Default is Basic Authentication.
Basic authentication request header name	The <code>x-introspect-basic-authorization-header</code> is available to provide a user-configured HTTP Basic authorization header.

Field	Description
<b>API Gateway only</b> Basic authentication username (optional)	The default user name for HTTP Basic authentication.
<b>API Gateway only</b> Basic authentication password (optional)	The default password for HTTP Basic authentication.
<b>API Gateway only</b> Token validation	Specifies the method used to determine the success of the introspection request that is sent to the third party service to validate the provided token. Select one of the following options: <ul style="list-style-type: none"> <li>Connected: The query is successful if the status return code is <b>200</b>.</li> <li>Active (default): The query is successful if the status return code is <b>200</b> and the response JSON body includes the property <b>active</b>: <b>true</b>.</li> </ul>
<b>API Gateway only</b> Custom header pattern (optional)	A regular expression for request headers that are to be passed to the third-party provider; for example, <code>x-Introspect-*</code> .
<b>API Gateway only</b> Authorization header pass through	Select this check box if you want to retain the <b>Authorization</b> header for a bearer token. The default behavior is to remove this header.

c. Enter the scopes in the third screen. A scope becomes an option in the request and response for an access token. Click Add to add additional fields for scopes. Click Next when done.

Field	Description
sample_scope_1	Scope for token
sample_scope_2	Scope for token
additional scopes	Scope for token

d. Review the settings on the Summary panel.

3. Click Save and Edit to complete the configuration.

## OAuth concepts for API Connect

OAuth is a token-based authorization protocol that allows third-party websites or applications to access user data without requiring the user to share personal information.

Note: In a multi-node cluster, OAuth operations will fail if quorum is lost. Quorum requires that the number of active nodes is greater than 50% of the total number of nodes in the cluster.

- [OAuth user scenario](#)  
Potential users of OAuth with IBM® API Connect have a number of methods to secure their API. The following scenario provides an overview of the available options.
- [OAuth introspection for third-party OAuth providers](#)  
OAuth token validation can be offloaded to the third-party Open Authorization (OAuth)/Open ID Connect (OIDC) provider by using the Introspection URL. Clients can use a third-party OAuth or OIDC provider to obtain a token that is protected by IBM API Connect. IBM API Connect can use this feature along with the mentioned provider to protect access to the API.
- [OAuth external URL and authentication URL](#)  
You can use the Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.
- [Scope](#)  
Per IETF RFC 6749, the value of the scope parameter is expressed as a list of space-delimited, case-sensitive strings.
- [Tokens](#)  
Tokens can be managed using refresh tokens and revocation URLs. You can extend the life of tokens using refresh tokens. You can end the life of a token by specifying a revocation URL.
- [Authentication URL user registry](#)  
You can use an Authentication URL user registry to specify a REST authentication service that manages user authentication, and optionally provides additional metadata to be embedded in the token.
- [Custom forms for user security](#)  
You can create custom forms for the authorization and identity extraction phase of OAuth.
- [Securing an API with a JSON Web Token](#)  
There are two methods to secure your API with a JSON Web Token. You can use the **jwt-generate** command, or you can use a token that has been generated external to IBM API Connect.
- [Troubleshooting OAuth](#)  
You can find the answers to some common questions in the Developer Portal, or in support communities and forums in DeveloperWorks, on GitHub, or on YouTube.

## Related information

- [The OAuth 2.0 Framework](#)

## OAuth user scenario

Potential users of OAuth with IBM® API Connect have a number of methods to secure their API. The following scenario provides an overview of the available options.

## Scenario overview

In this scenario, Alice is a user of an application. Alice can grant permission for an application to access specific information about Alice in a third-party system that is using OAuth. Depending on the type of OAuth that is supported by the target service, Alice does not enter her user name and password into the application. Instead, the application receives an access token that represents her credentials (user name and password). The application can now access the information about Alice in the target system.

For example, Alice maintains a list of books that she reads on a service that is provided by mybooks.com. Following the purchase of a smartphone, Alice installs an application on her new phone to display the book details. The phone application wants to call an API provided by mybooks.com, which can access the information. The mybooks.com API is secured by using the OAuth protocol.

To access the book details, the application must complete a two-step process:

1. The application must first obtain permission from Alice.
2. The application then uses that permission to call the target service and obtain the list of books.

In the first step, the application typically directs Alice to the provider of the target service, mybooks.com. Alice provides her user name and password, and gives permission for the application to access her information. It is important that Alice trusts that she is providing her credentials to the provider of the target service and not to an untrusted proxy application. For example, by checking that the security certificate of the website where Alice enters her credentials matches what Alice expects from the provider of the target service.

The result of this step is the access token that the application can use to call the API. The application then generates the appropriately formatted OAuth request. For example, the Authorization header, or HTTP query parameters, which includes the access token, consumer key, and signature method that are required by OAuth. This OAuth request is used to invoke the API proxy operation.

## Scenario within API Connect

No changes to the definition of your API operation are required to support this scenario.

1. Alice grants permission for the application to access her information *before* the invocation of the API.
2. When the application provides the Authorization header, or query parameters, containing the OAuth details about the call to the operation endpoint, the header is automatically passed through to the target service without any additional configuration.

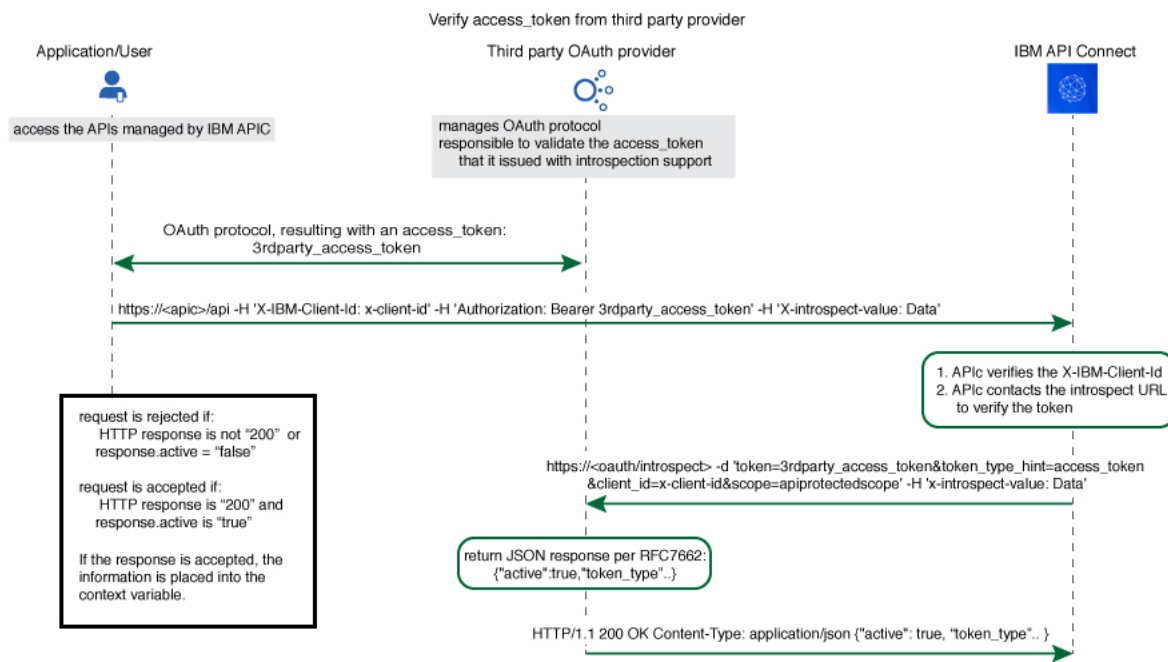
## OAuth introspection for third-party OAuth providers

OAuth token validation can be offloaded to the third-party Open Authorization (OAuth)/Open ID Connect (OIDC) provider by using the Introspection URL. Clients can use a third-party OAuth or OIDC provider to obtain a token that is protected by IBM API Connect. IBM® API Connect can use this feature along with the mentioned provider to protect access to the API.

You can use API Connect to protect an API that is secured by using the third-party OAuth access token in accordance with Introspection specification as defined in [RFC 7662](#). In addition to the specification, the `x-Introspect-` header is provided to pass other content to the third party as you require.

Authentication information can be carried in the request by configuring a basic authentication request header.

The following sequence diagram depicts the overall flow of request and response. The purpose of this diagram is only to provide a general visualization of the nature of the flow; for the precise details, refer to the explanatory information after the diagram.



The Introspect URL is configured in the Third Party OAuth provider configuration. See [Configuring a third-party OAuth provider](#).

When an API is protected by a third-party OAuth provider, API Connect will extract the bearer token and issue an HTTP POST request to the endpoint specified in the Introspect URL field.

The GET request is protected by API Connect with this feature.

You can use a header prefix to pass information to the third-party provider. The header prefix can include a regular expression and specifies the name pattern of the headers to use for sending additional information, such as `x-introspect-*`.

```
GET /petstore/pet/123 HTTP/1.1
Host: apiconnect.com
x-Introspect-type: dog
x-Introspect-name: simon
x-custom-apic: petstore123
Authorization: Bearer tGzv3JOkF0XG5Qx2T1KWIA
x-IBM-Client-Id: xxx-xxx
```

However, if you want to use a different header prefix, specify the required value in the Custom header pattern field in the third party OAuth provider configuration. For third-party OAuth provider configuration details, see [Configuring a third-party OAuth provider](#). The Custom header pattern feature is available only with the DataPower® API Gateway, if you are using the DataPower Gateway (v5 compatible) the header prefix must be `x-Introspect-`.

API Connect will issue this POST request to the introspection endpoint specified in `x-tokenIntrospect`, as illustrated in the code sample:

```
POST /oauth/introspectURL HTTP/1.1
Host: apiconnect.ibm.com
Content-Type: application/x-www-form-urlencoded
x-Introspect-type: dog
x-Introspect-name: simon
x-custom-apic: petstore123
token_type_hint=access_token&token=tGzv3JOkF0XG5Qx2T1KWIA
```

The third party OAuth/OIDC provider will respond with HTTP 200 indicating the request was successful and that payload contains the token information. API Connect honors the active claim as defined in the RFC specification.

If the value of the active claim is `true`, the token is treated as valid.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
```

```
{
 "active": true,
 "token_type": "bearer",
 "client_id": "xxx-xxx",
 "username": "John Smith",
 ...
}
```

If the value of the active claim is `false`, the token is treated as invalid.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
```

```
{
 "active": false
}
```

A response code other than HTTP 200 indicates failure to process the request.

When the OAuth token is valid and active, context variables are populated with information from the introspect JSON response. For more information, see [API Connect context variables](#).

When contacting the introspection endpoint, API Connect uses `client_id/appId` and `client_secret/appSecret` to construct the HTTP Basic authorization header.

By default, if `x-introspect-basic-authorization-header` exists in the request, the value is used for the HTTP Basic authentication header when the introspection endpoint is contacted. API Connect verifies that the HTTP Basic authentication header value is Base64 encoded before it is sent, and encodes it if necessary as shown in the following example. If the header is already encoded, it is sent without modification.

```
GET /petstore/pet/123 HTTP/1.1
Host: apiconnect.com
x-introspect-basic-authorization-header: user:password
```

API Connect issues the following:

```
POST ..
Authorization: Basic M3JkLXBhcncR5LWNsaWVudF9pZDozcmQtcGFydHlfY2xpZW50X3N1Y3JldA==token_type_hint=access_token&token= ..
```

If you are using the DataPower API Gateway, you can specify your own value for the HTTP Basic authentication header by providing the required value in the Basic authentication request header name field in the third party OAuth provider configuration. The following rules determine what authentication data is passed to the introspection endpoint:

- If there is a basic authentication header in the request, the specified credentials are used. The value must be either a string in the format `user:password`, or the Base64 encoded equivalent; API Connect will Base64 encode the value if necessary before sending the request to the introspection endpoint.
- If there is no basic authentication header in the request but there are values specified in the Basic authentication username and Basic authentication password fields in the third party OAuth provider configuration, those values are used.
- If there is no basic authentication header in the request, and user name and password details are not supplied in the third party OAuth provider configuration, but `client_id` and `client_secret` values are supplied in the body of the request, these are used.
- Otherwise, an error is returned.

For third-party OAuth provider configuration details, see [Configuring a third-party OAuth provider](#).

If either, or both, of scope and `scope validate url` are configured, and if the response is an active token with a scope claim from the third-party OAuth Provider introspection endpoint, API Connect will further enforce the scope validation in the following order:

1. If **scope** is configured for the OAuth API protection, verify the third-party scope against the scope that is configured.
2. If **scope validation url** is configured, verify the third-party scope against the scope validation URL.
3. If the third-party OAuth provider introspection endpoint does not return a scope, the Gateway will not check the scope defined in the security definition of the API where the third-party OAuth provider resource is used. For details on configuring an OAuth security definition in an API, see [Creating an OAuth security definition](#).

For more information, see [Scope](#).

**DataPower Gateway (v5 compatible) only**

By default the API Connect client ID and scope are sent to the third party OAuth provider. You can suppress this behavior in either of the following ways:

- Supply a **suppress-parameters** header as follows:

```
suppress-parameters: client_id
```

```
suppress-parameters: scope
```

or

```
suppress-parameters: client_id scope
```

depending on which parameters you want to suppress.

- Define an API property called **suppress-parameters** in the API definition itself, with one of the following string values:

```
client_id
```

```
scope
```

or

```
client_id scope
```

depending on which parameters you want to suppress. For information on how to define API properties, see [Setting API properties](#).

---

## OAuth external URL and authentication URL

You can use the Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.

### Including custom metadata in a token

In many scenarios, custom metadata needs to be included during the access token generation process. The metadata is either stored inside the access token or it is sent along with the access token to the client application. The client application can then send that access token, or the metadata in the payload, in a subsequent request to IBM® API Connect where the metadata is retrieved, validated, or sent to the downstream systems as required.

Examples include, but are not limited to:

- When resource owners are authenticated, metadata about the authenticated resource owner needs to be stored within the access token.
- The grant type that was used to obtain the token is another example of metadata stored within the access token.
- A confirmation code that needs to be sent to the client application along with access token is stored as metadata within the access token.

### Configuring External URL or Authentication URL in API Connect to obtain metadata

Metadata can be set by using either or both of the following URLs:

- External URL - When you call the External URL, an HTTP GET request is sent and API Connect expects an HTTP 200 OK along with an optional set of the specified response headers.
- Authentication URL - When you call the Authentication URL, the API Connect gateway sends a GET request with HTTP headers and then processes any HTTP response from the URL. For authentication, a REST authentication service is expected at the Authentication URL.  
See: [Authentication URL](#).

The External URL endpoint is entered in the Metadata section when you configure an OAuth Provider in the Cloud Manager or API Manager UI.

Use the following HTTP headers in the response, depending on the type of gateway you are using:

- DataPower® API Gateway:

```
X-API-OAUTH-METADATA-FOR-PAYLOAD
```

```
X-API-OAUTH-METADATA-FOR-ACCESSTOKEN
```

Note: These are the default header names for the DataPower API Gateway but you can override them; see [Configuring metadata for a native OAuth provider](#).

- DataPower Gateway (v5 compatible):

```
API-OAUTH-METADATA-FOR-PAYLOAD
```

```
API-OAUTH-METADATA-FOR-ACCESSTOKEN
```

The response header value from **X-API-OAUTH-METADATA-FOR-PAYLOAD** or **API-OAUTH-METADATA-FOR-PAYLOAD** is placed in the response payload and indicated as **metadata**.

The response header value from **X-API-OAUTH-METADATA-FOR-ACCESSTOKEN** or **API-OAUTH-METADATA-FOR-ACCESSTOKEN** is placed within the access token and indicated as **miscinfo**.

The two metadata response headers are case insensitive and you must escape any special characters in the string value content.

An example response payload that contains metadata along with the access token:

```
{
 "token_type": "bearer",
 "access_token": "AAEkNzhjDHYyyYy...cL0Mv6ct137w7ZU",
 "metadata": "m:metadata-for-payload_content"
 "expires_in": 3600,
 "scope": "read",
 "refresh_token": "AAEnj5SynCMYbF...oEZ6JjxYax_HdNg",
}
```

This example output from the token introspection endpoint shows the contents of the access token with `miscinfo` containing the metadata information.

```
{
 "active": true,
 "token_type": "bearer",
 "client_id": "78c2f10f-799a-4e1f-8e0a-098634997a35",
 "username": "Fred Smith",
 "sub": "fred",
 "exp": 1479850049,
 "expstr": "2016-11-22T21:27:29Z",
 "iat": 1479846449,
 "nbf": 1479846449,
 "nbfstr": "2016-11-22T20:27:29Z",
 "scope": "read",
 "miscinfo": "m:metadata-for-accesstoken_content",
 "client_name": "MobileApp"
}
```

## Input to the External URL

The following request headers are sent to the External URL.

Note: Any existing metadata values that were previously sent from the Authentication URL are also sent in the two request headers `x-existing-metadata-for-payload` and `x-existing-metadata-for-access-token`. The Metadata URL can make use of this information to create a new set of metadata values. The two request headers that are sent to the Metadata URL are displayed in bold text.

```
X-existing-metadata-for-payload payload-from-auth-url
X-existing-metadata-for-access-token token-from-auth-url
X-URI-in /org/env/miscinfo/oauth2/token (the URL that was sent to APIConnect for this particular token request)
X-METHOD-in POST
X-POST-Body-in client_id=client_id&grant_type=password&scope=read&username=name&password=password
X-X-Client-IP IP_address
X-X-Global-Transaction-ID ID_number
...
```

Note: If you are using the DataPower API Gateway rather than the DataPower Gateway (v5 compatible), the `X-POST-Body-in` header is not supported.

## Retrieving Metadata in API Connect

As described in the previous example scenarios, the metadata can be retrieved from the access token in the Application API and sent to the downstream systems. Retrieval can be done in the API assembly, which is secured to accept tokens in the security definitions.

In the resource API that accepts an access token, the `miscinfo` field can be accessed in the Assembly with the `oauth.miscinfo` context variable, as in the example.

```
apim.setvariable('message.body', apim.getvariable('oauth.miscinfo'));
```

You can also use token introspection to look at the contents of the access token.

## Refresh tokens and metadata

The Authentication URL (if configured for authentication) is invoked first, during authentication of the resource owner. The External URL is invoked as the last step, just before the generation of the access token. The only exception is when access tokens are generated from a refresh token. In cases where refresh tokens are used to generate new access tokens, the External URL is not invoked. The metadata from the refresh token is retained and then copied into the newly generated access token.

## Identifying the source of the metadata

The metadata is prefixed with keywords to indicate whether it originated from the External URL or the Authentication URL.

- Metadata from the External URL is prefixed with `m`:
- Metadata from the Authentication URL is prefixed with `a`:

Note: When revocation is enabled, some internal details are also stored in the `miscinfo` field, in brackets within the access token as shown in the following example:

```
"miscinfo": "[tlsprofile@https://api-revoke-url:443/server/revocation-url]m:metadata-for-accesstoken_content"
```

## Maximum size of the metadata

Metadata for the access token cannot exceed 512 bytes.

Metadata for the payload does not have a specific size restriction, except for when you use the Authorization code grant type. These restrictions are described in following sections.

## Characters are not allowed in metadata in certain scenarios

When you use the Authorization code grant type, or when a consent form is used for implicit grant type, there is a temporary state or code where the metadata from the authentication URL is stored. API Connect internally uses two prefixes - **!ma** and **!mp** to differentiate between payload and token metadata received from the Authentication URL and store them internally in the temporary state/code. Hence these specific character sequences - **!ma** and **!mp** should not be used as the metadata itself.

## Grant types and metadata

The OAuth grant types described in the following sections include **authorization code** (access code), **implicit**, and **client credentials** (application).

Authorization code grant type

- When metadata is included from an Authentication URL for an Authorization code grant type, as it is a three legged flow, both the content and the payload are stored within the **dp-state** and carried on to the authorization code and to the access token. Note that around 10 characters are used internally to differentiate between the metadata for payload and metadata for the access token when stored in the **dp-state**. In addition, if revocation is enabled, that will also be part of the token metadata. Hence the combined size of the token metadata, the payload metadata (including the 10 characters of internal data), and internal revocation details, cannot exceed 512 bytes in total.

If the overall size of the metadata exceeds 512 bytes, then the access token generation succeeds, but the metadata fields contain an error message of "metadata too large" as shown in the example.

```
"metadata": "m:error: metadata too large for AZ code grant type[Authorization Code-metadata-url-payload]"
"miscinfo": "[r:gateway]m:error: metadata too large for AZ code grant type[Authorization Code-metadata-url-token]"
```

This size restriction can be overcome when the metadata is sent from the Metadata URL and not from the Authentication URL, because the metadata is not stored in **dp-state** or in the authorization code.

Example of the **authorization code** (access code) grant type:

```
$ curl -k -d
"grant_type=authorization_code&code=$mycode&client_id=$myid&scope=scope_introspect&redirect_uri="
https://9.70.153.90/fei/sb/introspectpl/oauth2/token
{ "token_type": "bearer",
 "access_token": "AAEkOTH1ZDhhNjYtYTQ1ZS00YTMzLWE0N2QtZmE2OGZkMzQ0NzQ2OZn5Tl_TqYFeIfB7BFf6HFgibEoOjWXXEA84oFsWuE4NY-
RRZVdnGSaXAIYJz7s5vczfk5EV-BIb_6P_1YKm3ahrfrR5kI3sPO0uADEoseIP5-O9anUpEM5yhsayXvZbJ_6VDYz-hyXSJHTNqVj-
PHBialoRKBD5qca6k00fV2M", "metadata": "a:[Authorization Code-Test-auth-url-payload]", "expires_in": 3600,
 "scope": "scope_introspect", "refresh_token": "AAFAGlEVMbwicr_L0FTZ4q6HZ-
RcQygnIXFC9zbsFO4wd3hcniC4KQX21X0fL2c8cnmzCZgws8xxLzNyfjQhUJNG15C1GbIe3dwhXJdiWA0Go-
dduhVtCbG26sJRRXyRMeRwWnJsy1BETPI8HQEN4a_D7fmxKcTVRZBvq86byg95qelZKyERi0Lhxdd_O4Nvss" }

$ curl -k -H "X-IBM-Client-Id: $myid" -H "X-IBM-Client-Secret: $mysecret" -d
"token_type_hint=access_token&token=$mytoken" https://9.70.153.90/fei/sb/introspectpl/oauth2/introspect
{ "active": true, "token_type": "bearer", "client_id": "98ed8a66-a45e-4a33-a47d-fa68fd344746", "username": "anyuser",
 "sub": "anyuser", "exp": 1484766368, "expstr": "2017-01-18T19:06:08Z", "iat": 1484762768, "nbf": 1484762768,
 "nbfstr": "2017-01-18T18:06:08Z", "scope": "scope_introspect", "miscinfo": "[r:gateway]a:[Authorization Code-Test-auth-
url-token]", "client_name": "oauth_app" }
```

Implicit grant type

- When implicit grant type is used, the access token and the metadata are returned in the **Location** header as a fragment, as you see in the example.

```
< Location:
https://localhost#access_token=AAEkOTH1ZDhhNjYtYTQ1ZS00YTMzLWE0N2QtZmE2OGZkMzQ0NzQ2b2uS2KfWd-
nYBJSi4nmPxQBTLael17tKBPRMzwP5BC386nlxpoOTE1G748ZVH6Mq_TJL3GeV3PtXTVIkWOLBji_7t1jiQfpnVfrNkovvZkhUexYmFkcDsmLSdaWxZ6Pc
IMPAC4ojT8qV1sYV-
ChTk36yqOx_NiKimZaDikDk7WTg&expires_in=3600&scope=scope_introspect&token_type=bearer&metadata=a%3A[Implicit-Test-
auth-url-payload]

$ curl -k -H "X-IBM-Client-Id: $myid" -H "X-IBM-Client-Secret: $mysecret" -d
"token_type_hint=access_token&token=$mytoken" https://9.70.153.90/fei/sb/introspectpl/oauth2/introspect
{ "active": true, "token_type": "bearer", "client_id": "98ed8a66-a45e-4a33-a47d-fa68fd344746",
 "username": "anyuser", "sub": "anyuser", "exp": 1484768365, "expstr": "2017-01-18T19:39:25Z", "iat": 1484764765,
 "nbf": 1484764765, "nbfstr": "2017-01-18T18:39:25Z", "scope": "scope_introspect", "miscinfo": "[r:gateway]a:[Implicit-
Test-auth-url-token]", "client_name": "oauth_app" }
```

Client credentials grant type

Authentication URL will not be invoked when using client credentials grant type, as there is no resource owner. The metadata from Authentication URL is not available for this grant type. However, content returned from Metadata URL will be included as metadata.

## Behavior when retrieving metadata with both an External URL and an Authentication URL

If an External URL is configured and a connection to the external server is successful, the response headers overwrite any existing metadata obtained from the Authentication URL to become the final value. Therefore, you must carefully examine the incoming request headers and create appropriate response headers from the External URL.

If an External URL is configured, but the connection to the External URL fails, then a failure message of "error on metadata url" is written for metadata in both the payload and the access token.

If an External URL is configured and the connection is successful, but the remote server does not send any of the specified HTTP response headers, a blank value is written for metadata in both the payload and the access token.

Attention: An External URL overwrites existing values from Authentication URL. This includes blank values.

If no External URL is configured, the metadata that is obtained from the Authorization URL is retained as the final value.



# Scope

Per IETF RFC 6749, the value of the scope parameter is expressed as a list of space-delimited, case-sensitive strings.

In IBM® API Connect, scopes have no inherent meaning. Instead, scopes are defined in the OAuth Provider so that an application can request an access token that is valid for one or more of the scopes. In the secured API, scopes are listed as requirements for an access token to be considered valid. All scopes that are listed by the security definition for the API must be granted by the access token.

Note:

- An OAuth Provider must have at least one scope.
- An OAuth token will not be granted unless a default scope is configured at the provider, or one or more predefined scopes are included in the grant request.
- If no default scope is defined, and the request does not contain a scope, an invalid scope error is returned for the request.

## OAuth provider

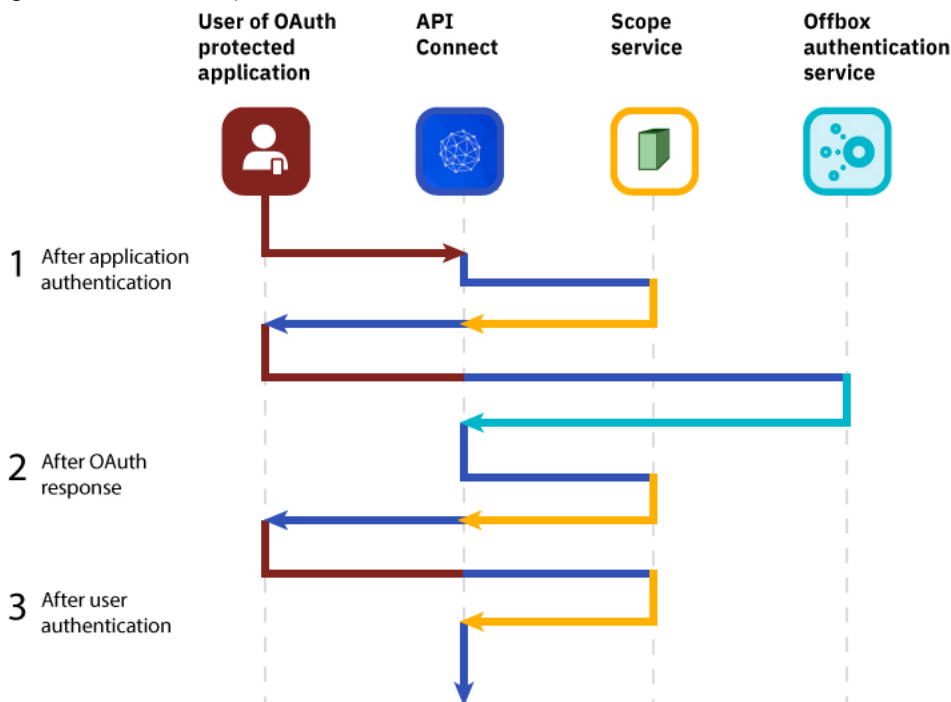
To provide more refined support for the OAuth scope handling, API Connect allows the [Authentication URL user registry](#) extension to modify the scope value.

When you define an [OAuth provider](#), the Advanced scope check extensions provide the flexibility to check and override allowed scopes. The optional extensions are Application scope check and Owner scope check.

The scope that is eventually received by the application is determined by the interactions that are described in the three following paragraphs. Scope processing follows the sequence of items 1, 2, and 3 in the following list, offering three opportunities to override the scope value. Figure 1 provides an overview of the process.

1. After the application successfully authenticates, and if OAuth Native provider  $\geq$  Advanced scope check  $\geq$  Application scope check is configured, API Connect makes a call to allow extra verification and uses the contents of `x-selected-scope` to override the scope value that was initially requested by the application. When Application scope check is enabled, the HTTP response header `x-selected-scope` must be present, or the call fails.
2. If OAuth Native provider  $\geq$  User Security  $\geq$  Authentication is configured to authenticate application users using an Authentication URL, then API Connect makes a call, as documented in [Authentication URL user registry](#). When the response code is `HTTP 200`, and the response header `x-selected-scope` is present, the value that is configured in `x-selected-scope` is used as the new scope value, overriding both what the application already requested and what was provided in the Application scope check described in paragraph 1. In the response header, `x-selected-scope` is an optional element.
3. After the user successfully authenticates, and if OAuth Native provider  $\geq$  Advanced scope check  $\geq$  Owner scope check is enabled and configured with a valid URL, API Connect makes a call to allow the content of `x-selected-scope` to refine the scope value. When Owner scope check is enabled, the HTTP response header `x-selected-scope` must be present, or the call fails.

Figure 1. OAuth advanced scope overview



The final scope permission that is granted by the access token is the result of the flow described in items 1, 2, and 3. [Figure 2](#) shows a more detailed view of the transaction flow with examples that show when `x-selected-scope` provides a new scope value.

Figure 2. OAuth advanced scope detail



```

OAuthProvider.yaml
oauth2:
 client-type: confidential

 scopes:
 scope:
 saving: Saving Account
 checking: Checking Account
 scope-validators:
 application:
 url: https://scope-service/application
 owner:
 url: https://scope-service/owner

 authentication:
 x-ibm-authentication-url:
 url: https://offbox-authenticate-service/authenticate

```

initiate an OAuth  
 GET /oauth2/authorize?client\_id=88&scope=saving&state=xyz..

If client\_id is not valid, reject.

scope:saving

```

GET /application?app-name={AppName}&appid={AppId}&
org={OrgName}&orgId={OrgId}&catalog={Catalog}&
catalogid={CatalogId}&transid={TransactionId}&
token_scope=saving&api_scope=saving%20checking
Host: scope-service

```

HTTP/1.1 200 OK  
 x-selected-scope: premier

If scope service does not return x-selected-scope,  
 API Connect uses the original requested scope (saving).

scope:premier

prompt user for credential  
 username=alice, password=secret

```

GET /authenticate
Host: offbox-authenticate-service
Authorization: Basic YWxpY2U6c2VjcmV0
X-Global-Transaction-ID: {TransactionId}

```

HTTP/1.1 200 OK  
 x-selected-scope: premier chk-888  
 api-authenticated-credential: cn=alice,o=spoon

scope:premier chk-888

```

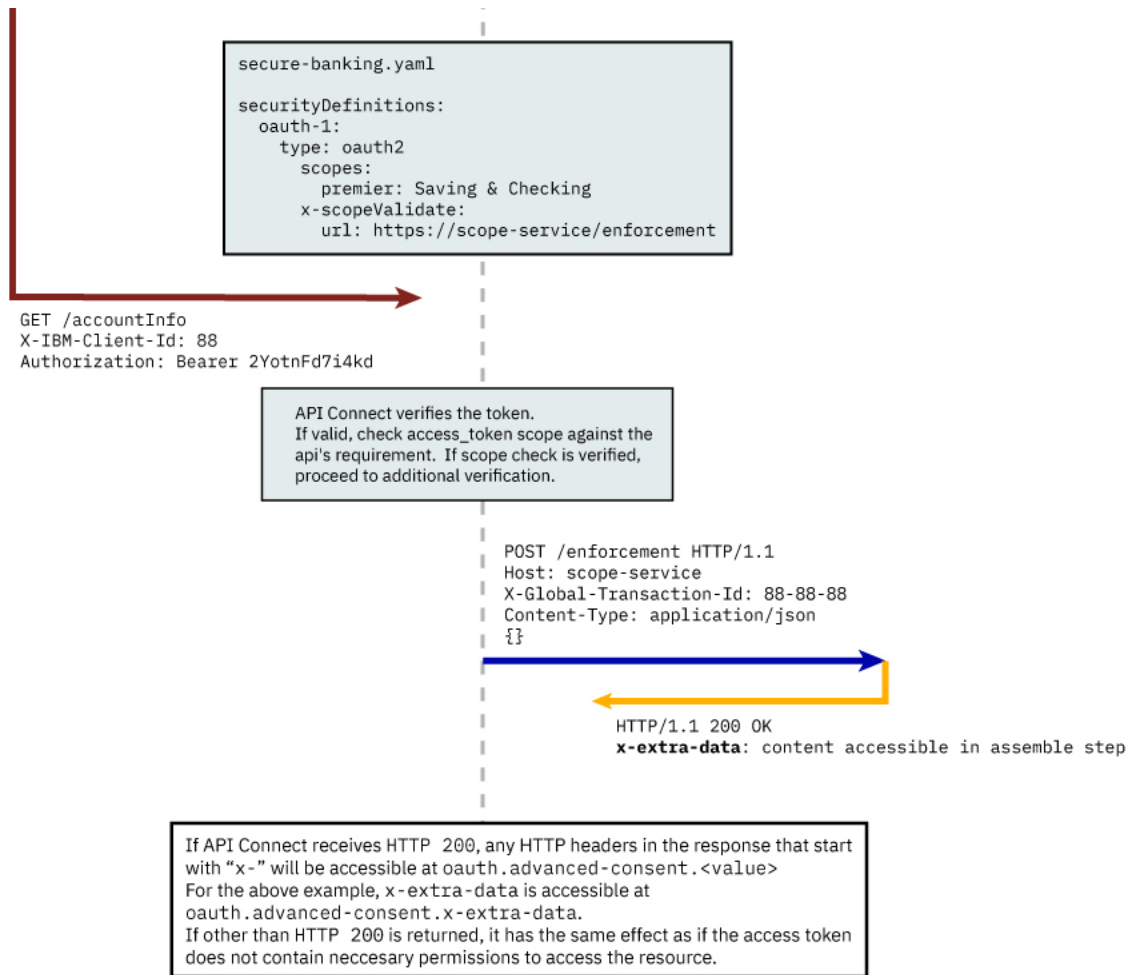
GET /owner?app-name={AppName}&appid={AppId}&org={OrgName}&
orgId={OrgId}&catalog={Catalog}&catalogid={CatalogId}&
transid={TransactionId}&token_scope=premier%20chk-888&
api_scope=saving%20checking
Host: scope-service

```

HTTP/1.1 200 OK  
 x-selected-scope: premier chk-888 gold

scope:premier chk-888 gold

HTTP/1.1 200 OK  
 {"access\_token": "2YotnFd7i4kd",  
 "token\_type": "bearer",  
 "scopes": "premier chk-888 gold"}



## Consumer API enforcement

### Standard scope validation

To access the API `/getaccount` the application must send a `GET` request with an access token that contains the scope, or scopes, defined in the OAuth provider. If the request doesn't contain a scope, you must specify a default scope to use. If no default scope is defined, and the request doesn't contain a scope, an invalid scope error is returned for the request.

```

GET /getaccount
HTTP/1.1
Host: server.example.com
X-IBM-Client-Id: 8888-8888-8888
Authorization: Bearer AAEkNjVkOWIyYjgtOWY5ZS00YWQwLWIyYzktZ

```

The following application OpenAPI file `secure-banking.yaml` defines the scope, or scopes, that must exist in the token to be granted access to the API `/getaccount`.

```

secure-banking.yaml

securityDefinitions:
 scope-only:
 type: oauth2
 description: ''
 flow: implicit
 authorizationUrl: ''
 scopes:
 checking: 'Checking Account'
 saving: 'Saving Account'
 mutual: 'Mutual Fund Account'
 security:
 - scope-only:
 - checking
 - scope-only:
 - saving
 - mutual

```

In the examples, the access token `AAEkNjVkOWIyYjgtOWY5ZS00YWQwLWIyYzktZ` is able to access the API because it contains one, or a combination of `scope-only` that is defined in `secure-banking.yaml` such as:

- `checking`
- `saving mutual`
- `checking saving mutual`

Advanced Scope Check

Administrators can enable an additional scope check by configuring the consumer API property Advanced Scope Check URL that becomes `x-scopeValidate` as shown in the following OpenAPI file example.

```
securityDefinitions:
 advanced-scope-only:
 type: oauth2
 description: ''
 flow: implicit
 authorizationUrl: ''
 scopes:
 checking: 'Checking Account'
 saving: 'Saving Account'
 mutual: 'Mutual Fund Account'
 x-scopeValidate:
 url: 'https://advanced-scope-check.bk.com/validate-scope'
 tls-profile: 'ssl-client'
```

After API Connect successfully verifies the access token against any scope requirement, API Connect will make an **HTTP POST** request to the `x-scopeValidate` endpoint similar to the following example. Response code **HTTP 200** from the endpoint indicates a success. Any other response code, or a connection error, is treated as a permission error for the token.

```
POST /validate-scope?app-name=..&appid=..&org=..&orgid=..&catalog=..&catalogid=..&transid=..
HTTP/1.1
Host: advanced-scope-check.bk.com
Content-Type: application/json
```

```
{
 "context-root" : checking,
 "resource" : accountinfo,
 "method" : GET,
 "api-scope-required" : [jointaccount],
 "access_token" : {
 "client_id" : "2cd71759-1003-4a1e-becb-0474d73455f3",
 "not_after" : 174364070,
 "not_after_text" : "2017-07-11T02:27:50Z",
 "not_before" : 174360470,
 "not_before_text" : "2017-07-11T01:27:50Z",
 "grant_type" : "code",
 "consented_on" : 1499736470,
 "consented_on_text" : "2059-07-11T01:27:50Z",
 "resource_owner" : "cn=spoon,email=spoon@poon.com",
 "scope" : "jointaccount mutual",
 "miscinfo" : "[r:gateway]"
 }
}
```

An example of successful response follows.

```
HTTP/1.1 200 OK
Cache-Control: no-store
Pragma: no-cache
X-Custom-For-Assemble-Process: audit
```

Any HTTP response header that begins with "x-" is kept as the context variable `oauth.advanced-consent`. Based on the example successful response, `X-Custom-For-Assemble-Process: audit` becomes `oauth.advanced-consent.x-custom-for-assemble-process`, and can be accessed in the assemble step.

Additional validation options

You can optionally use additional fields for validation:

Request Header

Defines the regular expression to match against **request** headers. Matching headers are included in the request to the advanced scope validation endpoint.

Response Context Variable

Defines the regular expression to match against **response** headers. Matching headers are saved as context variables in the format `oauth.advanced-consent.*`.

## Related information

- [IETF RFC 6749](#)

## Tokens

Tokens can be managed using refresh tokens and revocation URLs. You can extend the life of tokens using refresh tokens. You can end the life of a token by specifying a revocation URL.

This section contains information regarding token management, including refreshing and revoking tokens, redirecting, and managing tokens with the DataPower Gateway.

- [Refresh tokens](#)  
If you are using OAuth authentication, you can enable refresh tokens. Refresh tokens are issued to the client to obtain a new access token when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or narrower scope.
- [Token revocation](#)  
In IBM® API Connect, you can revoke or refresh tokens. If necessary, you can also revoke all tokens that are issued to a specific client ID or a resource owner.
- [Authenticating and authorizing through a redirect URL](#)  
You can use a service that is hosted externally from IBM API Connect to collect authentication and authorization details from your user when an application requests access on that user's behalf.
- [Token management with the DataPower Gateway \(v5 compatible\)](#)  
API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.

- [Token management with the DataPower API Gateway](#)

API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.

---

## Refresh tokens

If you are using OAuth authentication, you can enable refresh tokens. Refresh tokens are issued to the client to obtain a new access token when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or narrower scope.

When you are using OAuth authentication, API requests must include a valid access token, by using the Authorization HTTP header. Access tokens that are issued by the IBM® API Connect Token Endpoint are valid for 3600 seconds (1 hour) by default, as indicated by the `expires_in` property that is returned on the token request. The following code block shows an example API request with an Authorization header:

```
GET /bankingApi/accountSummary?client_id=32427ce5-bb7c-48a7-9de3-4bb629091103
HTTP/1.1
Accept: application/json
Host: api.ibm.com
Authorization: Bearer AAEFYy1hbGx1hdS5nVX4x6iTL2sb3ymBivQb...
```

After an access token expires, if the option is enabled in the OAuth provider configuration Tokens > Refresh tokens screen, the application uses refresh tokens. Each refresh token is valid for approximately 31 days after it is issued (or for the Time to Live time period specified) and can be used only once to request a new access token. Along with the new access token, a new refresh token is also returned. For details on how to enable refresh tokens, see [Configuring a native OAuth provider](#).

If the access token is expired and the application does not have a refresh token, it must restart the OAuth exchange by using the choice of Grant Type(s) allowed by the OAuth provider.

---

## Token revocation

In IBM® API Connect, you can revoke or refresh tokens. If necessary, you can also revoke all tokens that are issued to a specific client ID or a resource owner.

---

### Token revocation using an external third party service

This topic describes token revocation using a third party, external service. This option is configured in the Native OAuth provider, using the Token Management > Type = Third party screen. The revocation URL is an endpoint that links to an external service which contains information about access or refresh tokens. API Connect is involved in the initial creation and validation of tokens. When an OAuth revocation URL is present, API Connect calls the URL to determine if the associated token can be trusted. The token server then checks a token *blocklist* (a data store of inactive tokens) to ensure that the token is still valid. If the token is still valid, API Connect continues the processing.

---

## Examples

A number of revocation examples follow. The first shows a sample fetch request and the response from a remote revocation URL.

---

### GET request and response

In this example, an API Gateway server issues a GET request to the Revocation URL and receives a result. It shows that different resource owners (Laura and Emily) can revoke all tokens when using the same application (client ID).

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
GET <revocationURL> HTTP/1.1
User-Agent: IBM-APIManagement/4.0
Accept: application/xml; text/xml
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-revocation>

<!--
Access Tokens and/or Refresh Tokens that are revoked can be individually listed.
To keep this list small, please only include access tokens and refresh tokens that are valid.
For access tokens, any token older than 20 minutes is no longer valid.
For refresh tokens, any token older than 44700 minutes is no longer valid.
-->

<token type="access">AAETb2F1dGgtcmV2b2t1LWN1c3RvbfZaR1VbnPScl...</token>
<token type="refresh">fZaR1VbnPSclUGTjCRdq4mPbOosD2+aZIKbJ6bTeW...</token>

<!--
If a resource owner has revoked all tokens issued to a given application, please
list them as shown here.
-->

<resource-owner client-id="760d75a2-44b1-4485-8c6f-0d264fcf7398">laura</resource-owner>
<resource-owner client-id="760d75a2-44b1-4485-8c6f-0d264fcf7398">emily</resource-owner>

</oauth-revocation>
```

## POST request and response

---

The next example shows a post request and response.

Request:

```
POST <revocationURL> HTTP/1.1
User-Agent: IBM-APIManagement/4.0
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<token>
 <token_type>bearer</token_type>
 <access_token>AAENYy1hbGwtcmVmcmVzaOfNeQKX8ZeojsBY9v0FI7/OerQvzKHq...</access_token>
 <expires_in>3600</expires_in>
 <scope>3600</scope>
 <resource-owner>alice</resource-owner>
 <client_id>83d9cdcd-ba72-4d00-abae-005da8da5fb1</client_id>
</token>
```

Response:

```
HTTP/1.1 200 OK
```

## Provide token information on revocation request

---

In this example, the application calls an API and passes a bearer token. In response, the Gateway fetches the revocation URL and provides information on the token being verified.

Gateway:

```
GET <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2t1LWN1c3RvbfZaRlVbnPScl
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Alternatively, the same process occurs when using a refresh token to issue a new access token. The application sends a refresh request to the token service. The Gateway then fetches the revocation URL, providing information on the refresh token being verified.

Gateway:

```
GET <revocationURL>GET HTTP/1.1
refresh-token: AAETb2F1dGgtcmV2b2t1LWN1c3RvbfZaRlVbnPScl
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Note: If you are using a third party OAuth provider then, for API Calls with bearer tokens, when Introspect URL is enabled on the API, the revocation URL is not applicable. Instead, the third party endpoint must validate the token and also check for revocation before returning a **200 OK** response to the Gateway.

## Revoking tokens issued to Alice up to and including a specific date

---

On May 1st, Alice loses her phone and needs to reset her password. As a result, the token provider wants to revoke every token issued before Alice lost her phone. In this example, the Gateway sends a **GET** request to the revocation service. The revocation service replies confirming revocation of the specified tokens.

Gateway:

```
GET <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2t1LWN1c3RvbfZaRlVbnPScl
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Revocation Service:

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-revocation>
 <resource-owner before="2015-05-01T09:30:10Z">alice</resource-owner>
</oauth-revocation>
```

## Revoking all tokens issued up to and including a specific date

---

Under certain catastrophic conditions, you may need to revoke all tokens issued up to and including a specific date, for example, May 1st. In this example, the Gateway sends a **GET** request to the revocation service. The revocation service replies confirming revocation of the specified tokens.

Gateway:

```
GET <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2t1LWN1c3RvbfZaRlVbnPScl
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Revocation Service:

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-revocation>
 <everytoken before="2015-05-01T09:30:10Z" />
</oauth-revocation>
```

## Putting it all together

The following shows the examples contained in this topic executed in a single action:

Gateway:

```
GET <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2t1LWN1c3RvbfZaRlVbnPSc1
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Revocation Service:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Cache-Control: public, max-age=120
Date: Fri, 08 May 2015 21:49:03 GMT
```

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-revocation>
 <resource-owner before="2015-04-08T09:30:10Z">mary</resource-owner>
 <resource-owner before="2015-04-12T09:30:10Z">john</resource-owner>
 <resource-owner before="2015-04-13T09:30:10Z">kevin</resource-owner>
 <resource-owner before="2015-04-01T09:30:10Z">alice</resource-owner>
</oauth-revocation>
```

Notes:

- In the previous example, there are no entries older than one month in the response (the maximum life of a refresh token).
- Each response is cached for up to two minutes according to response's directive.
- The **before** attribute uses the **xs:dateTime** syntax.

## Authenticating and authorizing through a redirect URL

You can use a service that is hosted externally from IBM® API Connect to collect authentication and authorization details from your user when an application requests access on that user's behalf.

### Before you begin

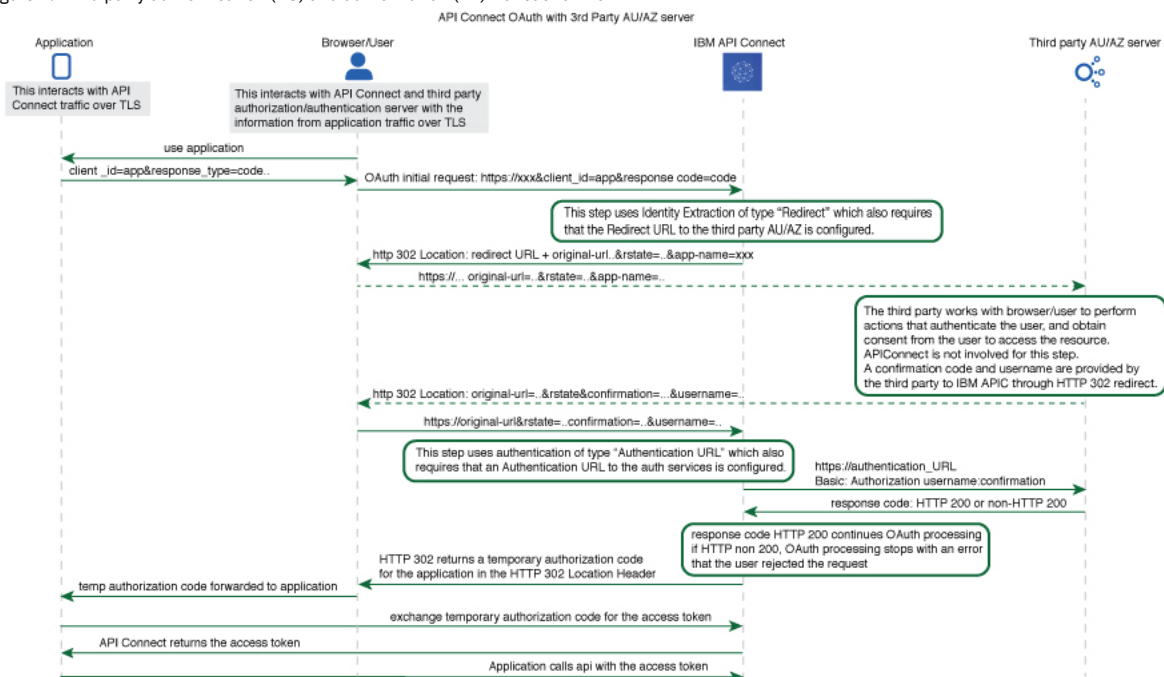
To complete this task, you will need to either create or have created an OAuth security definition that uses Implicit grant type or Access Code (Authorization Code) grant type. For more information, see [Creating an OAuth security definition](#).

### About this task

If you use methods for authentication that are not supported by API Connect, you can redirect users to a suitable URL at which they can authenticate. The user is then returned to the OAuth process after authentication and authorization have been confirmed.

The following illustration indicates the transaction flow for third party authentication.

Figure 1. Third party authentication (AU) and authorization (AZ) transaction flow



1. The application initiates a request to access an API protected with a third-party entity. API Connect redirects the application with an **HTTP 302** redirect based on **identity extraction** -> **redirect** -> **redirect-url**, for user authentication (and optional authorization).
  2. The application communicates directly with the third-party entity to gather user identity. API Connect is not involved in this communication. After the third-party entity finishes processing authentication (and optional authorization), it returns an **HTTP 302** redirect that uses the **original-url** from the request, with the username and confirmation code appended.
  3. API Connect receives the request that includes the username and confirmation code, and communicates with the authentication URL, based on **authentication** -> **x-ibm-authentication-url**, to confirm user identity before the request is completed.
  4. An **HTTP 200** response from the third-party entity allows API Connect to continue the OAuth 2.0 request process as if the owner is authenticated. The request is then processed according to the **grants** type.
    - **accessCode** returns a temporary code to the application.
    - **implicit** returns the access token to the application.
- For any response other than HTTP 200, the request fails with a statement added to the error log.

## Procedure

To create an external form, and to indicate the URL to which API Connect will redirect users, complete the following instructions:

1. Create your service for authentication and authorization. You will use the URL of the landing page of this service as your redirect URL.
  - a. To include elements in your form that are provided by API Connect, use the following query parameters from the URL that your user is redirected to. Note: With the exception of **original-url**, none of these parameters are included in the URL automatically; you must add them as required.

**app-name**  
The name of the application requesting access, as provided through the Developer Portal.

**appid**  
The id of the application requesting access.

**catalog**  
The name of the catalog where the product is being used by the application.

**catalogid**  
The id of the catalog where the product is being used by the application.

**catalogtitle**  
User-friendly display name for the catalog.

**org**  
The name of the consumer organization that hosts the application.

**orgid**  
The id of the consumer organization that hosts the application.

**orgtitle**  
User-friendly display name for the organization.

**original-url**  
The original URL that the user was directed to by the application, including query parameters from the original URL that are necessary for standard OAuth 2.0 requests. You can include these parameters in your service to provide information to the user. Additionally the **state\_nonce** is appended. The **state\_nonce** is a hash code generated by API Connect for verification purposes. The URL is URL-encoded and should be decoded before further use, the **state\_nonce** should remain unchanged.

**provider**  
The name of the API provider organization.

**providerid**  
The id of the API provider organization.

**providertitle**  
User-friendly display name for the provider organization.

**requested-scope**  
[optional] If [Application Scope check](#) is enabled and replaces the **scope** from the initial application request, this field holds the **scope** value from the initial application request, and the new replacement scope value is put into **original-url**.

**transid**  
transaction id used in the GW for the transaction which trigger this call

The URL to which the user is sent to when they are redirected to your page has the following form:

**Redirect\_URL?original-url=Original\_URL&state\_nonce=R\_State&app-name=Application\_Name**

where all variables are as described previously. The Redirect URL does not have a size limit enforced by API Connect.

- b. Create the stages of authentication, authorization, and any intermediate stages that you require to take place before you allow access to the application. Upon completion of these stages, redirect the user to the *Original\_URL* and append a user name, their confirmation code, and the application name to be evaluated for access grant or denial by API Connect. The confirmation code does not have a size limit enforced by API Connect. Original URL requires the following form:

**Original\_URL?username=User\_Name&confirmation=Confirmation\_Code**

where all variables are as described previously.

For example:

**https://your\_gateway.com/your\_org/your\_catalog/your\_api/oauth/authorize?response\_type=code&redirect\_uri=https://example.com/redirect&scope=/your\_api&client\_id=5af57a4a-6db9-4141-ad08-5709432af66e&state\_nonce=HoIbRG+6bZtq1B7LDkq4gj1D3SHKq1CbnYdHs/bMz2Y=&username=spoon&confirmation=12345678**

- c. To send your own error responses after the authentication and authorization service, redirect the user to the *Original\_URL* and append an error code. You can also append a user name. Use the following form:

**Original\_URL?username=User\_Name&error=Error\_Response**

where *Error\_Response* is the message you wish to send and all other variables are described as previously.



For example:

```
https://your_gateway.com/your_org/your_catalog/your_api/oauth/authorize?
response_type=code&redirect_uri=https://example.com/redirect&scope=/your_api&client_id=5af57a4a-6db9-4141-ad08-
5709432af66e&state_nonce=HoIbRG+6bZtq1B7LDkq4gj1D3SHKglCbnYdHs/bMz2Y=&username=User&error=access_denied
```

2. Create a service to validate the confirmation code and user name. API Connect makes a GET call to your authentication URL after the user is redirected back to the authorization URL. When the call is made, it includes in its authorization header the user name and confirmation code you supplied previously. Confirm that these are correct and respond with an HTTP success code such as 200 OK if you want to allow access, or non-200 HTTP response code, such as 401 Unauthorized to deny access.
3. In your OAuth provider configuration, supply the redirect URL that is used in Step 1 and the authentication URL that is used in Step 2.  
For more information on configuring an OAuth Provider, see [Configuring a native OAuth provider](#) when using API Manager or [Configuring a native OAuth provider](#) when using Cloud Manager.

---

## Token management with the DataPower Gateway (v5 compatible)

API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.

In order to manage tokens with the DataPower® Gateway (v5 compatible), you must set the Token Management Type to Native in your Native OAuth provider configuration. See [Configuring token management and revocation for a native OAuth provider](#) when using Cloud Manager or [Configuring token management and revocation for a native OAuth provider](#) when using API Manager.

Also, the DataPower quota enforcement server must be enabled on the DataPower Gateway to use the distributed cache to manage tokens. See [Quota enforcement](#).

When distributed cache support is enabled, replay protection is provided across the gateway cluster through the quota enforcement server. This support ensures that the same token cannot be reused across the members of the quota enforcement peer group.

Important:

If you have a cluster of DataPower Gateway servers, the OAuth data synchronization behavior across the servers depends on whether or not you enable revocation:

- If you enable revocation, API Connect uses the DataPower quota enforcement server, and OAuth data is synchronized across the servers. If an access token is obtained from one server, the OAuth data synchronization ensures that the same authorization code cannot be used to obtain an access code from another server. You must ensure that the DataPower quota enforcement server is configured.
- If you disable revocation, API Connect does **not** use the DataPower quota enforcement server and OAuth data does not synchronize across the cluster of DataPower Gateway servers. Therefore, to prevent the same authorization code being used to obtain an access code from more than one server you must configure DataPower to synchronize OAuth data across the servers, by using the DataPower Gateway console.

To configure DataPower to synchronize OAuth data, ensure that the DataPower quota enforcement server is configured. For more information, see [Configuring the quota enforcement server](#).

---

## Resource owner revocation

When Resource owner revocation path is selected in the Token Management screen, the configuration inserts two REST API calls to /oauth2/issued.

- An HTTP GET operation that retrieves a list of all granted permissions for a specific user.
- An HTTP DELETE operation that revokes an application for a specific user.

The setting inserts header-based security definitions of client ID and client secret as shown in the **View permissions example**. The API call to revoke a given application for a given user is shown in the **Revoke permissions example**.

View permissions example

To list all the applications granted by user `cn=spoon,o=ibm` with username `spoon` and password `spoon` using a registered administration application of `5287fe53-8747-438a-8262-681ec75b79c5`.

- Request:

```
GET /oauth2/issued
HTTP/1.1
Host: apic.ibm.com
x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
Authorization: Basic c3Bvb246c3Bvb24=
```

- Response:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
[
 {
 "clientId": "7369ad66-5674-b7d3-4567-de35283421aca",
 "owner": "cn=spoon,o=ibm",
 "clientName": "PetStore Application",
 "scope": "listpet",
 "issuedAt": 1503327054,
 "consentedOn": 1503327054,
 "expiredAt": 1503330654,
 "refreshTokenIssued": false,
 "appId": "d2031f0f27339315333734ab9",
 "org": "PetStoreOrg",
 "orgTitle": "Katie Pet Grooming Inc",
 "orgId": "5887803de4b06e6998c4b2c7",
 "provider": "SuperStore",
 "providerTitle": "Simon SuperStore",
```

```

 "providerId": "5887803de4b06e6998c4b2c7",
 "catalog": "publicapi",
 "catalogTitle": "For public",
 "catalogId": "5887803de4b06e6998c4b2d3"
 },
 {
 "clientId": "a8746323-9825-a842-8736-abd8202356ac8",
 "owner": "cn=spoon,o=ibm",
 ...
 }
]

```

Revoke permissions example

To revoke application `a8746323-9825-a842-8736-abd8202356ac8` by owner `cn=spoon,o=ibm`.

- Request

```

DELETE /oauth2/issued?client-id=a8746323-9825-a842-8736-abd8202356ac8
HTTP/1.1
Host: apic.ibm.com
x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
Authorization: Basic c3Bvb246c3Bvb24=

```

- Response

```

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache

{ "status": "success" }

```

## Client revocation

When Client revocation path is selected in the Token Management screen, the result is the following:

This option inserts one REST API call to `/oauth2/revoke`, which supports OAuth 2.0 [IETF RFC 7009](https://tools.ietf.org/html/rfc7009). An HTTP POST operation that an application can send to this API to revoke either an `access_token`, or `refresh_token` with `token_type_hint` as shown in the following examples:

Revoke `access_token`

- Request:

```

POST /oauth2/revoke
HTTP/1.1
Host: apic.ibm.com
x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
Content-Type: application/x-www-form-urlencoded

token_type_hint=access_token&token=AAIHZGVmYXVsdD1-KqwD0Yc3EDn941SWX14xuR...

```

- Response:

```

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache

{ "status": "success" }

```

Revoke `refresh_token`

- Request:

```

POST /oauth2/revoke
HTTP/1.1
Host: apic.ibm.com
x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
Content-Type: application/x-www-form-urlencoded

token_type_hint=refresh_token&token=.....

```

- Response:

```

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache

{ "status": "success" }

```

## Token management with the DataPower API Gateway

API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.

Note: The steps described in this topic are required if you are using the DataPower® OVA or appliance version of the gateway. For the API Connect supplied Kubernetes deployment, the API Security Token Manager is preconfigured provided that, in the gateway service custom resource file, v5 compatibility mode is disabled and the token management service is enabled:

```
spec:
 .
 .
 .
 .
 .
 .
 .
 apicGatewayServiceV5CompatibilityMode: false
 tokenManagementService:
 enabled: true
 .
 .
 .
```

For more information, see [Installing the DataPower Gateway subsystem](#).

In order to manage tokens with the DataPower API Gateway, you must set the Token Management Type to Native in your Native OAuth provider configuration. See [Configuring token management and revocation for a native OAuth provider](#) when using Cloud Manager or [Configuring token management and revocation for a native OAuth provider](#) when using API Manager.

Note: This setting is also required when you use a third-party OAuth provider and `Introspect` `cache type` is set to `protocol` or `Time to live`.

For token management with the DataPower API Gateway, you must configure the API Security Token Manager on the gateway. Complete the following steps:

1. Log in to the DataPower administration console, selecting apiconnect for the domain, and WebGUI for the Graphical Interface.
2. In the search box, enter API Security Token Manager, then click on the API Security Token Manager link that displays in the search results.
3. For the Administrative state, select enabled.
4. Click the + icon alongside the Gateway Peering field to create a new gateway peering object.
5. Provide a Name and a Local address.
6. In the Local port and Monitor port fields, provide values that aren't already in use by services for the Local address.
7. Ensure that Peer group mode is selected.
8. Alongside the Peers field, use the add button and accompanying field to add at least two peers, to ensure that quorum is achieved.
9. If the Enable SSL option is selected, select a value for the Identification credentials.
10. For Persistence location, select a setting other than memory.
11. When done, click Apply.
12. Repeat steps 1 to 11 for each DataPower API Gateway in the peer group.
13. When done, click Apply, then click Save Configuration to save your changes.

For more information, see [Defining the API security token manager](#) and [Creating a gateway peering instance](#).

## Resource owner revocation

When Resource owner revocation path is selected in the Token Management screen, the configuration inserts two REST API calls to /oauth2/issued.

- An HTTP GET operation that retrieves a list of all granted permissions for a specific user.
- An HTTP DELETE operation that revokes an application for a specific user.

The setting inserts header-based security definitions of client ID and client secret as shown in the **View permissions example**. The API call to revoke a given application for a given user is shown in the **Revoke permissions example**.

View permissions example

To list all the applications granted by user `cn=spoon,o=ibm` with username `spoon` and password `spoon` using a registered administration application of `5287fe53-8747-438a-8262-681ec75b79c5`.

- Request:

```
GET /oauth2/issued
HTTP/1.1
Host: apic.ibm.com
x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iN1uU6bt5cV4dN7nW5kM5uP8vL3uF3cT7
Authorization: Basic c3Bvb246c3Bvb24=
```

- Response:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
[
 {
 "clientId": "7369ad66-5674-b7d3-4567-de35283421aca",
 "owner": "cn=spoon,o=ibm",
 "clientName": "PetStore Application",
 "scope": "listpet",
 "issuedAt": 1503327054,
 "consentedOn": 1503327054,
 "expiredAt": 1503330654,
 "refreshTokenIssued": false,
 "appId": "d2031f0f27339315333734ab9",
 "org": "PetStoreOrg",
 "orgTitle": "Katie Pet Grooming Inc",
 "orgId": "5887803de4b06e6998c4b2c7",
 "provider": "SuperStore",
```

```

 "providerTitle": "Simon SuperStore",
 "providerId": "5887803de4b06e6998c4b2c7",
 "catalog": "publicapi",
 "catalogTitle": "For public",
 "catalogId": "5887803de4b06e6998c4b2d3"
 },
 {
 "clientId": "a8746323-9825-a842-8736-abd8202356ac8",
 "owner": "cn=spoon,o=ibm",
 ...
 }
]

```

Revoke permissions example

To revoke application `a8746323-9825-a842-8736-abd8202356ac8` by owner `cn=spoon,o=ibm`.

- Request

```

DELETE /oauth2/issued?client-id=a8746323-9825-a842-8736-abd8202356ac8
HTTP/1.1
Host: apic.ibm.com
x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iN1uU6bt5cV4dN7nW5kM5uP8vL3uF3cT7
Authorization: Basic c3Bvb246c3Bvb24=

```

- Response

```

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache

```

```
{ "status": "success" }
```

## Client revocation

When Client revocation path is selected in the Token Management screen, the result is the following:

This option inserts one REST API call to `/oauth2/revoke`, which supports OAuth 2.0 [IETF RFC 7009](https://tools.ietf.org/html/rfc7009). An HTTP POST operation that an application can send to this API to revoke either an `access_token`, or `refresh_token` with `token_type_hint` as shown in the following examples:

Revoke `access_token`

- Request:

```

POST /oauth2/revoke
HTTP/1.1
Host: apic.ibm.com
x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iN1uU6bt5cV4dN7nW5kM5uP8vL3uF3cT7
Content-Type: application/x-www-form-urlencoded
token_type_hint=access_token&token=AAIHZGVmYXVsdD1-KqWd0Yc3EDn94lSWX14xuR...

```

- Response:

```

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache

```

```
{ "status": "success" }
```

Revoke `refresh_token`

- Request:

```

POST /oauth2/revoke
HTTP/1.1
Host: apic.ibm.com
x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
x-ibm-client-secret: E2qM6mG2bX2uClxT2iN1uU6bt5cV4dN7nW5kM5uP8vL3uF3cT7
Content-Type: application/x-www-form-urlencoded
token_type_hint=refresh_token&token=.....

```

- Response:

```

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache

```

```
{ "status": "success" }
```

## Authentication URL user registry

You can use an Authentication URL user registry to specify a REST authentication service that manages user authentication, and optionally provides additional metadata to be embedded in the token.

This support can optionally enable any of the following:

- Providing the authenticated credential to IBM® API Connect. For example, the user logs-in with user name: `spoon`, and password: `fork`. When the user is authenticated, the credential becomes `cn=spoon,o=eatery`. The credential is kept in the OAuth `access_token` to represent the user.
- Providing metadata support. Allow extra metadata to be stored in the `access_token`.
- Overriding the `scope` that the application receives after a successful OAuth protocol processing. By responding with a specific header, the Authentication URL endpoint can replace the `scope` value that the application receives. For example, you can provide a specific resource owner an account number within the `scope` header response for use in future processing steps.

When you call the Authentication URL user registry, the API Connect gateway sends a GET request with HTTP headers and then processes any HTTP response from the URL. For authentication, a REST authentication service is expected at the Authentication URL.

The following response from the REST authentication service indicates that user authentication is successful and that API Connect will use `cn=spoon,o=eatery` as the user identity.

```
HTTP/1.1 200 OK
Server: example.org
X-API-Authenticated-Credential: cn=spoon,o=eatery
```

For information on how to configure a User Security policy in an API assembly for use with an Authentication URL user registry, see [User Security policy](#).

For an example of an OAuth provider configuration that uses an Authentication URL user registry, see [Example - using multiple OAuth policies in an OAuth provider assembly](#).

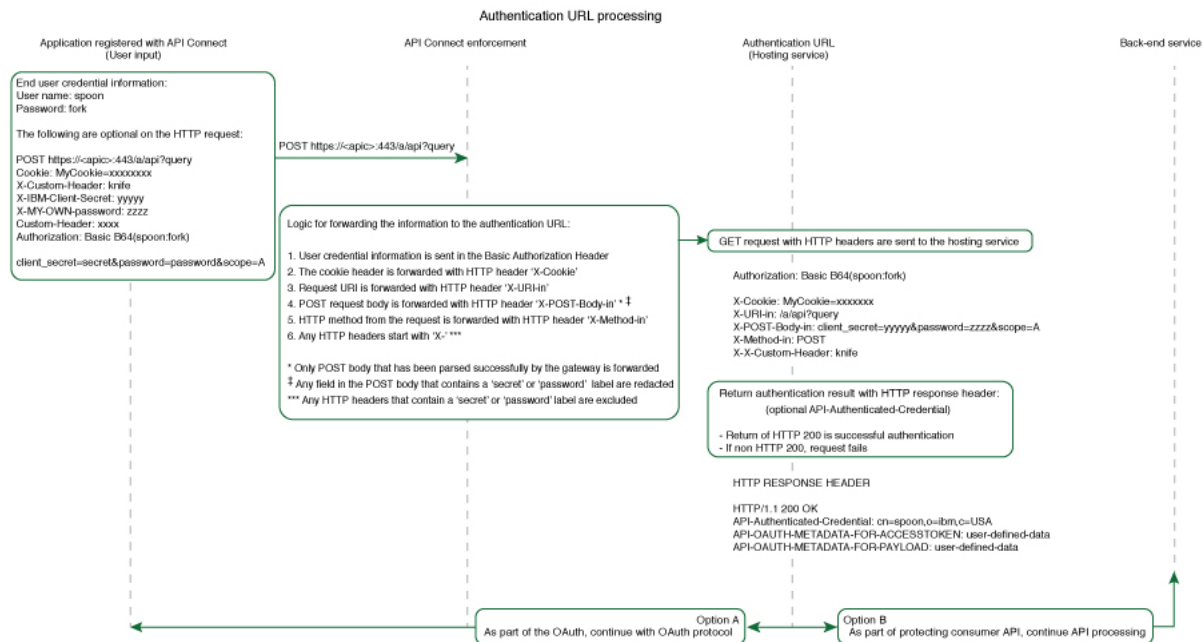
API Connect considers any non-200 HTTP response code a failed user authentication attempt.

When an Authentication URL user registry is invoked, two HTTP response headers are available that include metadata in the access token or the response payload that contains the access token. For more information, see [OAuth external URL and authentication URL](#). The two metadata response headers are:

```
API-OAUTH-METADATA-FOR-ACCESSTOKEN
API-OAUTH-METADATA-FOR-PAYLOAD
```

When an Authentication URL is invoked, an HTTP response header is available to override the requested `scope` from the application. For more information, see [Scope](#). The response header is:

```
x-selected-scope
```



If you are using the DataPower® API Gateway rather than the DataPower Gateway (v5 compatible), this diagram is provided for guidance only and is not fully accurate for this release.

## Custom forms for user security

You can create custom forms for the authorization and identity extraction phase of OAuth.

### About this task

The Native OAuth provider configuration provides the capability for requiring additional authentication and authorization steps for user security. Custom HTML forms may be created to extract the identity of the user and to authorize the user. This capability applies to Implicit, Resource owner password, and Access code grant types.

- [Creating a custom HTML login form for user security](#)  
Custom HTML forms can be created for user security during the identity extraction stage in OAuth.
- [Creating a custom HTML authorization form for user security](#)  
Custom HTML forms can be created for user security during the authorization stage in OAuth.

## Creating a custom HTML login form for user security

Custom HTML forms can be created for user security during the identity extraction stage in OAuth.

### Before you begin

The Native OAuth provider configuration includes Identity Extraction when using the Implicit, Access code, or Resource owner password grant types. You have the option to select how to extract the user credential and one of the choices is Custom HTML Form. For more information, see [Configuring a native OAuth provider](#) when using API Manager or [Configuring a native OAuth provider](#) when using Cloud Manager. For more information, see [Configuring a native OAuth provider](#). This topic describes how to create the Custom HTML form for identity extraction.

### About this task

During three-legged OAuth definitions (Implicit flow, Resource owner password flow, and Access (Authorization) code flow, the user is presented with a form for signing in to the service provided by the API. You can present a custom form or a default form. Your custom form must fulfill certain requirements.

Important: The fields used by IBM® API Connect to inject information into your form have case-sensitive field names.

### Procedure

To create a custom sign-in form for your Native OAuth provider, complete the following steps:

1. Create a well formed XHTML document that will be parsed and transformed by API Connect to inject hidden fields.
2. For your XHTML form, set the method as POST, the encoding type as application/x-www-form-urlencoded, and the action as authorize. Add any other parameters that you require.  
For example:

```
<form method="POST" enctype="application/x-www-form-urlencoded" action="authorize">
```

3. Create a text input field that is named username and create a password input field named password.
4. Add the line <EI-INJECT-HIDDEN-INPUT-FIELDS>. This third element is a placeholder that API Connect replaces with input fields to complement the user-submitted data.
5. Create a button to initiate the sign-in process.  
For example:

```
<button id="home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.apionprem.doc_oauth_custom_login_form_2_login_button" type="submit" name="login" value="true">Log in</button>
```

6. Optional: Add text that is displayed the first time that the user visits the sign-in page. Use the tag <EI-LOGINFIRSTTIME> for the text that you want to display.
7. Optional: Add text that appears when the user is returned to the sign-in page if they fail to authenticate. Use the tag <EI-LOGINFAILED> for the text that you want to display.
8. Optional: Have an error message displayed when an error in the custom form prevents it from being displayed to the user correctly. Use the tag <EI-INTERNAL-CUSTOM-FORM-ERROR/>; the message text is generated automatically. You should detect such errors during testing to prevent this error message being displayed to the end user.
9. Optional: You can add elements that are loaded from external sources, such as images or JavaScript.
10. Insert spacing and other features as you require. Completing Steps 1 through to 8 results in a form similar to the following example:

```
<html lang="en" xml:lang="en">
 <head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/></head>
 <body>
 <form method="POST" enctype="application/x-www-form-urlencoded" action="authorize">
 <h1>Please sign in</h1>
 <p>Username </p>
 <p><input type="text" name="username" required="required" /> </p>
 <p>Password </p>
 <p><input type="password" name="password" required="required" /> </p>
 <EI-INJECT-HIDDEN-INPUT-FIELDS/>
 <p> <button
id="home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.apionprem.doc_oauth_custom_login_form_2_login_button" type="submit" name="login" value="true">Log in</button> </p>

 <EI-LOGINFIRSTTIME>
 <p>If you have forgotten your user name or password, contact your system administrator.</p>
 </EI-LOGINFIRSTTIME>

 <EI-LOGINFAILED>
 <p>At least one of your entries does not match our records.
 If you have forgotten your user name or password, contact your system administrator.</p>
 </EI-LOGINFAILED>

 <EI-INTERNAL-CUSTOM-FORM-ERROR/>

 </form>
 </body>
</html>
```

11. Make your form available at a URL of your choice.
12. If you have not already done so, configure your Native OAuth provider to use a Custom HTML form for identity extraction and provide the URL at which your form is available. For more information, see [Configuring a native OAuth provider](#) when using API Manager or [Configuring a native OAuth provider](#) when using Cloud Manager.

## Related tasks

- [Creating a custom HTML authorization form for user security](#)

## Related information

- [Configuring API security](#)

# Creating a custom HTML authorization form for user security

Custom HTML forms can be created for user security during the authorization stage in OAuth.

## Before you begin

The Native OAuth provider configuration includes user authorization when using the Implicit, Access code, or Resource owner password grant types. You have the option to select how to authorize application users and one of the choices is Custom HTML Form. For more information, see [Configuring a native OAuth provider](#) when using API Manager or [Configuring a native OAuth provider](#) when using Cloud Manager. This topic describes how to create the Custom HTML form for authorization.

## About this task

During three-legged OAuth definitions (Implicit flow, Resource owner password flow, and Access (Authorization) code flow, the user is presented with a form through which they grant permission to an application to access their data through the API on their behalf. You can present a custom form or a default form. Your custom form must fulfill certain requirements.

Important: The fields used by IBM® API Connect to inject information into your form have case-sensitive field names.

## Procedure

To create a custom authorization form for your Native OAuth provider, complete the following steps:

1. Create a well-formed XHTML document. This will be parsed and transformed by API Connect to inject hidden fields.
2. For your XHTML form, set the method as POST, the encoding type as application/x-www-form-urlencoded, and the action as authorize. Add any other parameters that you require.  
For example:

```
<form method="POST" enctype="application/x-www-form-urlencoded" action="authorize">
```

3. Add the line <AZ-INJECT-HIDDEN-INPUT-FIELDS/>. This line is a placeholder that API Connect will replace with input fields necessary for the completion of the OAuth process.

4. Create two buttons with the following code so that the user can grant or deny permission. Edit the text to suit your preferences.

```
<button class="cancel" type="submit" name="approve" value="false">No Thanks</button>
<button class="submit" type="submit" name="approve" value="true">Allow Access</button>
```

5. Optional: Display an error message when an error in the custom form prevents it from being displayed to the user correctly. Use the tag <AZ-INTERNAL-CUSTOM-FORM-ERROR/>; the message text is generated automatically. You should detect such errors during testing to prevent this error message being displayed to the end user.

6. Optional: You can add to the form HTML elements that will load features from external sources, such as images or JavaScript.

For example, <script src="http://www.example.com/example.js" />

7. Insert spacing and additional elements as you require. Completing Steps 1 through to 6 results in a form similar to the following example:

```
<html lang="en" xml:lang="en">
<head><title>Request for permission</title></head>
<body class="customconsent">
 <div>
 <div>
 <form method="post" enctype="application/x-www-form-urlencoded" action="authorize">
 <AZ-INJECT-HIDDEN-INPUT-FIELDS/>
 <p>Greeting..</p><DISPLAY-RESOURCE-OWNER/>
 <p>This app </p><OAUTH-APPLICATION-NAME/><p> would like to access your data.</p>
 <div>
 <button class="cancel" type="submit" name="approve" value="false">No Thanks</button>
 <button class="submit" type="submit" name="approve" value="true">Allow Access</button>
 </div>
 </form>
 </div>
 <AZ-INTERNAL-CUSTOM-FORM-ERROR/>
</body>
</html>
```

8. Make your form available at a URL of your choice.
9. If you have not already done so, configure your Native OAuth provider to use a Custom HTML Form for authorization for User Security. Provide the URL as the endpoint at which your form is available. For more information, see [Configuring a native OAuth provider](#) when using API Manager or [Configuring a native OAuth provider](#) when using Cloud Manager.

## Related tasks

---

- [Creating a custom HTML login form for user security](#)

## Related information

---

- [Configuring API security](#)

# Securing an API with a JSON Web Token

---

There are two methods to secure your API with a JSON Web Token. You can use the **jwt-generate** command, or you can use a token that has been generated external to IBM® API Connect.

## About this task

---

JSON Web Token (JWT) is an OAuth 2.0 compliant method of authentication that can be useful to secure your API in API Connect.

You can secure your API with a JSON Web Token by using either of the following methods:

- Generate a token through the **jwt-generate** command, and then augment the response payload with your generated token replacing the `id` token.
- Use a token that was generated outside of API Connect and include it into the response payload, by using the metadata URL.

## Procedure

---

Create a **jwt-generate** policy in the assembly.

- a. In API Manager, open the Assembly tab.
- b. Add a **jwt-generate** policy to the assembly for the API.

## Troubleshooting OAuth

---

You can find the answers to some common questions in the Developer Portal, or in support communities and forums in DeveloperWorks, on GitHub, or on YouTube.

You can use the following links to go to the topics:

- [OAuth 2.0 support in the developer portal test tool](#)
- [OAuth requires quorum in a multi-node cluster](#)

## OAuth 2.0 support in the developer portal test tool

---

OAuth 2.0 support is exposed as an API through the provider OpenAPI definition. You can use the test tool that is included with API Connect to test the OAuth 2.0 configuration. For more information, see [Testing an API using the Developer Portal test tool](#).

## OAuth requires quorum in a multi-node cluster

---

In a multi-node cluster, OAuth operations will fail if quorum is lost. Quorum requires that the number of active nodes is greater than 50% of the total number of nodes in the cluster.

## API Analytics

---

You can use API Connect to filter, sort, and aggregate your API event data. You can present the results within correlated charts, tables, and maps to help you manage service levels, set quotas, establish controls, set up security policies, manage communities, and analyze trends.

API analytics is built on the OpenSearch open source real-time distributed search and analytics engine.

The content of the API event data depends on the logging policy that is set for the operation. For more information about the fields that are displayed in an API event record, see [API event record fields](#). For information about how to configure your logging preferences for API events, see [activity-log.policy](#) and [Including elements in your assembly](#).

Use the following tasks to configure and manage analytics:

- [Catalogs, Spaces, and Analytics](#)  
Understand how syndication and the use of Catalogs and Spaces affects the use of the features in API Connect Analytics.
- [Accessing analytics](#)  
Analytics data is displayed in dashboards in the API Manager UI. Dashboards are scoped at the provider organization, Catalog, and Space level. Analytics are also accessible at the cloud level in the Cloud Manager UI, and using the Toolkit CLI and REST API.
- [Analytics dashboards](#)  
IBM API Connect analytics provides dashboards for viewing your analytics data.
- [Exporting API event data](#)  
Export your API Connect API event data using either the toolkit CLI or the REST API.



- [Accessing analytics data with the toolkit CLI](#)  
You can access your analytics event data by using the API Connect toolkit CLI.
- [Accessing analytics data with the REST API](#)  
You can access your Analytics event data by using the API Connect REST API.
- [Modifying analytics API event data](#)  
How to modify API event data before it is stored.

---

## Catalogs, Spaces, and Analytics

Understand how syndication and the use of Catalogs and Spaces affects the use of the features in API Connect Analytics.

The data for Analytics is collated from API events that are logged when API operations are invoked. Analytics data for an API is scoped to the Catalog or Space where that API resides, and is only included in search results (and thus, visualizations and dashboards) for the owning Catalog or Space.

Access to the analytics data, and to the analytics functions in the API Manager user interface, can be managed through the use of Catalogs and Spaces, and the roles and permissions that are assigned to the users (or *members*) of the provider organization.

The cloud admin user can view aggregated analytics data for all catalogs, spaces, in all provider organizations, in the admin UI.

---

## Catalogs, Spaces, and permissions

Catalogs act as deployment targets through which APIs (in their containing Plans and Products) are staged and published to consumer organizations. API Manager users in the provider organization can be assigned the following access to the Analytics component for a Catalog:

- A role that has the Analytics\_>View permission for a Catalog: These users can view the analytics data generated for the APIs in the Catalog within *dashboards*, export dashboard data in its raw format or as event records, and apply filters to the data shown within the dashboards.
- A role that has the Analytics\_>Manage permission for a Catalog: This is a legacy permission and has the same permissions as the View permission.

The IBM® API Connect syndication feature provides a way for you to partition a Catalog into multiple deployment targets (or *Spaces*) through which separate groupings of APIs (in their containing Plans and Products) can be staged and published. Each Space can be allocated to a separate group of users who need to manage their Products independently, and the analytics data in each Space is scoped to those Products only. API Manager users in the provider organization can be assigned the following access to the Analytics component for a Space:

- A role that has the Analytics\_>View permission for a Space: These users can view the analytics data generated for the APIs in the Space within dashboards, export dashboard data in its raw format, and apply filters to the data shown within the dashboards.
- A role that has the Analytics\_>Manage permission for a Catalog: This is a legacy permission and has the same permissions as the View permission.

For more information about assigning Catalog or Space permissions to a role, see [Managing Catalog membership](#) and [Managing Space membership](#).

For more information about the default API Manager roles and permissions, see [API Connect user roles](#).

---

## Accessing analytics

Analytics data is displayed in dashboards in the API Manager UI. Dashboards are scoped at the provider organization, Catalog, and Space level. Analytics are also accessible at the cloud level in the Cloud Manager UI, and using the Toolkit CLI and REST API.

---




## About this task

The ability to access analytics data at a Catalog or Space level depends on your assigned roles and permissions. For more information about Catalogs and Spaces, see [Working with Catalogs](#) and [Using syndication in API Connect](#).

Restriction: If the visualizations in your dashboards display no data or show data only for a time period in the past, a possible reason might be that access to analytics data within API Connect is switched off. Contact your administrator for confirmation. (For more information, see [Configuring analytics offload for API Connect](#).)

---

## Procedure


- To open the analytics dashboard at Catalog scope, complete the following steps:
  1. In the API Manager UI, select Manage  in the navigation pane.
  2. From the list of available Catalogs in the API Manager Dashboard, select the required Catalog.
  3. Click the Analytics tab.  
The analytics dashboard for the Catalog is displayed.
- To open the analytics dashboard at Space scope, complete the following steps:
  1. In the API Manager UI, select Manage  in the navigation pane.
  2. From the list of available Catalogs in the API Manager Dashboard, select the required Catalog.
  3. Click the Spaces tab.
  4. Select the Space that you want to view.
  5. Click the Analytics tab.  
The default analytics dashboard for the Space is displayed.
- To view analytics at Provider Organization scope, in the API Manager UI, select the analytics icon  from the navigation pane.
- To view analytics at the Cloud level, in the Cloud Manager UI: [Accessing cloud level analytics](#)
- You can also access Analytics data using the Toolkit CLI or the REST API: [Accessing analytics data with the toolkit CLI](#)

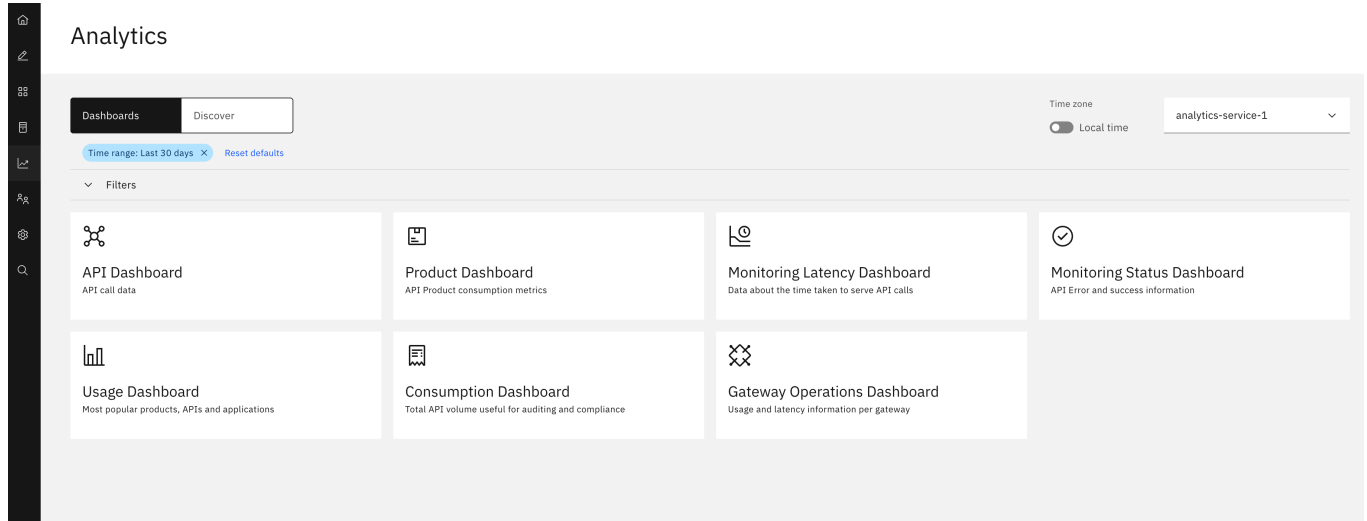
# Analytics dashboards

IBM® API Connect analytics provides dashboards for viewing your analytics data.

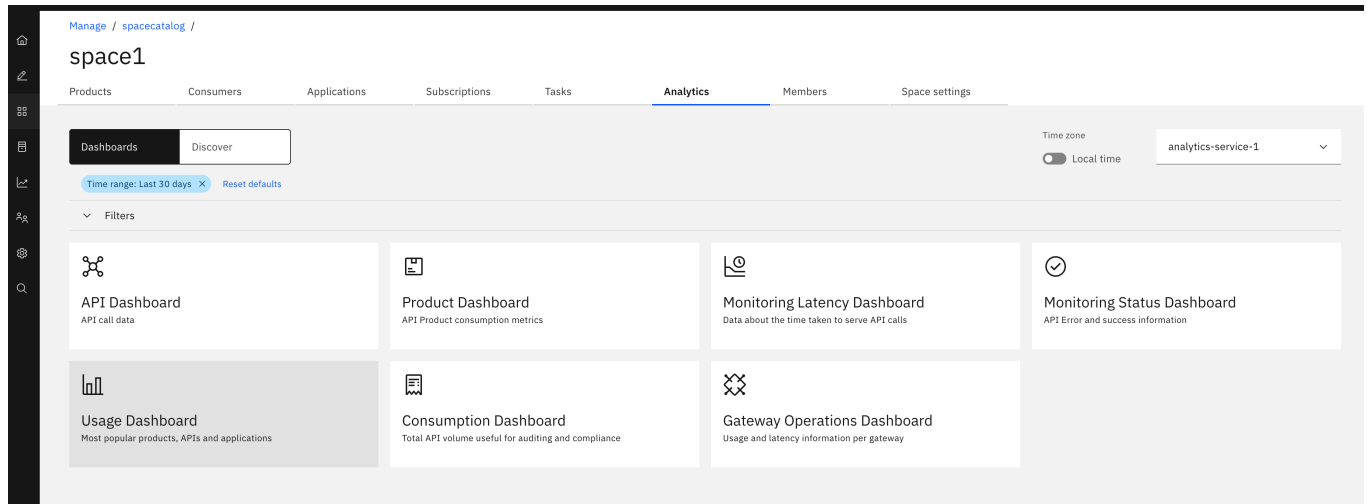
## Analytics dashboards

The analytics dashboards in the API Manager UI show the API event data at the scopes of provider organization, catalog, and space. The scope of the analytics data shown depends on how you access the analytics dashboards, see: [Accessing analytics](#).

The provider organization scoped dashboards are accessed from the settings icon  in the navigation pane:



The space scoped dashboards are accessed from the Analytics tab when viewing the space:

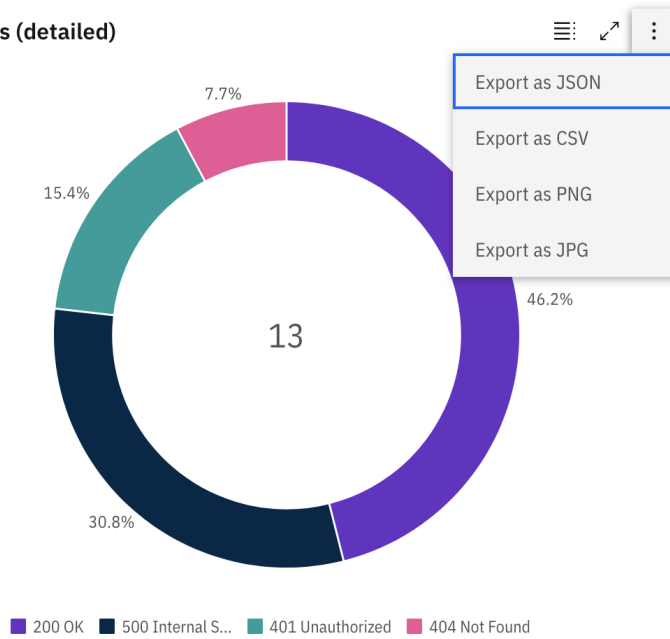


Seven dashboards are available:

- **API Dashboard**- contains charts that summarize total API calls, response codes, and response times.
- **Product Dashboard** - contains charts that show total API calls and application subscriptions per plan.
- **Monitoring Latency Dashboard** - contains charts that provide response time statistics and data usage.
- **Monitoring Status Dashboard** - contains charts that show the response codes and success/failure rates of API calls.
- **Usage Dashboard** - contains charts that show the top five APIs, products, and apps.
- **Consumption Dashboard** - contains charts that show total API consumption in your cloud.
- **Gateway Operations Dashboard** - contains charts that show API usage and latency by gateway.

Each chart in the dashboards can be exported as a PNG or JPG image. The source data for each chart can be exported as a JSON or CSV file. To export a chart, click

### Status codes (detailed)



Actions at the upper right:

It is also possible to enlarge the chart to full-screen by clicking . To see the source data in tabular form click .

## Discover view

By default the Dashboards tab is displayed. To view your API event data in tabular form, select Discover:

### Analytics

Time zone: analytics-service-1 (Local time)

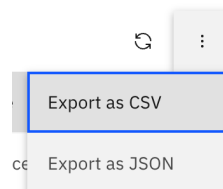
Time range: Last 30 days X Reset defaults

Filters

About 144 results (0.008 seconds) at 16:07:36 UTC+1

Date time	Status code	API name	API version	Method	Total time elapsed (ms)	Gateway IP	Transaction ID	App name
11/04/2023, 12:15:18	200 OK	rb-findbranch	2.0.0	GET	34	192.168.87.255	26496	andre-spacecatalog-app2
11/04/2023, 12:15:17	200 OK	rb-xmltojson	1.0.0	POST	4	192.168.87.255	26448	andre-spacecatalog-app2

The maximum number of API events that can be accessed from the UI is 10,000. The maximum number of event records that can be displayed per page on the Discover view is 200.



You can export the API event data from this view by clicking Actions at the upper right:

Note: When you export from the Discover view, only the data displayed on the current page of results is exported. If you want to export the data from all pages, you must export separately from each page. To export all your analytics data, do not use the UI, see [Exporting API event data](#).

## Filtering displayed data

You can filter the data that is displayed in the Dashboards and Discover view by defining queries. Expand the Filters section to view, define, and apply analytics queries to your displayed data:

Filters

New Saved Shared

Time range Load query  
Select one of the following time range filters

Last 15 minutes  Last 24 hours  
 Last 30 minutes  Last 7 days  
 Last 1 hour  Last 30 days  
 Last 4 hours  All events  
 Last 12 hours  Custom

Fields Add  
Select from the following dropdowns and enter a value

Field	Operator	Value	Delete
API name	equals	creditcheck	

Type  
Select one of the following type filters

All events  
 Success only (2xx, 3xx)  
 Errors only (4xx, 5xx)

Cancel Save as... Apply

You can save and share the queries that you define:

- Shared queries are visible to all users at the same scope. They are not visible to users in a different scope. For example, a query that is shared by a Cloud Manager UI user is not visible to an API Manager UI user. Within the API Manager UI, a query that is created at catalog scope is not visible to users viewing analytics data at provider organization scope.
- Other users cannot edit or delete queries that you share, but they can create a duplicate query, which they can edit and share.
- All queries have a name and an optional description. The name does not need to be unique.

Saved queries can be exported as strings for use in REST API and toolkit CLI queries, for example:

Filters

New Saved Shared

Search by name, description or query string

Name	Description	Last modified	Shared
<input checked="" type="radio"/> boo		11/04/2023, 16:16:03	--
<input type="radio"/> ccheck		11/04/2023, 13:39:11	

Edit  
Share  
Duplicate  
Copy to clipboard  
Delete

copies the following string to the clipboard:

```
api_name>equals:boo&timeframe=last30days
```

## Exporting API event data

Export your API Connect API event data using either the toolkit CLI or the REST API.

The export function in the [Analytics dashboards](#) exports just the data that is displayed in the UI. To export up to 10,000 API event records, you can use the toolkit CLI. To export all your analytics data use the REST API.

To export data with the toolkit CLI:

1. Follow the steps to install and configure the toolkit CLI: [Accessing analytics data with the toolkit CLI](#).
2. Use the **events** command to export your analytics data; for example to export the events for a provider organization:

```
apic -m analytics events:orgList --analytics-service <analytics_service> --server <management_server_api_endpoint> --format=json
```

```

"total": 200,
"search_time": 7,
"events": [
 {
 "log_policy": "activity",
 ...
 }
]

```

Your data is returned as a JSON array, each element is a single API event record.

To export data with the REST API:

1. Follow the steps to get started with the REST API: [Accessing analytics data with the REST API](#).
2. Call the `events` API. For example, to get all analytics API event records from a provider organization, specify the organization's API path:

```

curl -v -k -H 'Accept: application/json' -H 'Authorization: Bearer <bearer_token>' -X GET --url
'https://<management_server_api_endpoint>/analytics/<analytics_service>/orgs/<provider_organization>/events'
{
 "total": 45543,
 "search_time": 36,
 "events": [...]
}

```

To get just the events for a provider organization, specify the organization in the API path: `orgs/<provider_organization>/events`.

Note: To get just the events for a catalog, specify the catalog in the API path; for example:

```

/analytics/<analytics_service>/catalogs/<provider_organization>/<catalog>/events

```

- **API event record fields**

An API event is logged each time an API operation is invoked, and an event record is generated for each API event in the Gateway server. The API event record contains information about the API call and the content of the record depends on the logging policy that is set for the operation.

## API event record fields

An API event is logged each time an API operation is invoked, and an event record is generated for each API event in the Gateway server. The API event record contains information about the API call and the content of the record depends on the logging policy that is set for the operation.

You can use the activity-log policy to configure your logging preferences for the API event details that are stored in the Analytics component. By default, invocation details are logged if an API call is successful, and invocation, header, and payload (message body) details are logged if an API call results in an error code. To override these default settings and change the level of detail that is included in the API event record, you can add the activity-log policy to your API assembly and then configure the policy's properties. For example:

- To include details about the request body or response body in the API event record for a successful API call, you can add an activity-log policy to the associated API operation and set the content type to `payload`.
- To include details about the HTTP request headers or HTTP response headers in the API event record for a successful API call, you can add an activity-log policy to the associated API operation and set the content type to either `header` or `payload`.
- To ensure the value in the `client_geoip` field is accurate, the gateway must receive the `X-Forwarded-For` header in all API calls. Check with the administrator of your deployment environment to ensure that the `X-Forwarded-For` header is passed to your gateways. For example, in a Kubernetes environment where NINGX ingresses are used, configure the NGINX ingress that is used by your gateway to use the `X-Forwarded-For` header: [NGINX Configuration "use-forwarded-headers"](#).

For more information about how to configure your logging preferences, see [activity-log policy](#) and [Including elements in your OpenAPI 2.0 API assembly, Including elements in your OpenAPI 3.0 API assembly](#).

Tip: A log policy field is included in the event record to identify the logging setting. To see examples of the invocation, header, and payload details that can be included in an API event record, see:

- [Example: Event record with invocation details \(activity logging setting\)](#).
- [Example: Event record with invocation and header details \(header logging setting\)](#).
- [Example: Event record with invocation, header, and payload details \(payload logging setting\)](#).

The following table lists the static set of fields that are displayed in an API event record.

Table 1. API event record fields

Field name	Type	Description
@timestamp	Date	A timestamp that records when the record was written by the Logstash data collection engine that feeds data into OpenSearch.
api_id	String	The API identifier.
api_name	String	The name of the API.
api_version	String	The version number of the API.
app_id	String	The identifier for the registered application.
app_name	String	The name of the registered application. Note: The data indicates <b>N/A</b> when a client ID is not used or is invalid on the API. The gateway needs a client id to determine which app was invoking the API. From this app, the gateway can determine what plan the app is subscribed to on the product that contains the API. Without a client ID, the gateway is unable to determine what plan, product, or app was invoked because a single API can belong to multiple products (each of which have multiple plans and apps subscribed to those plans with client IDs).
app_type	String	The application type, with a value of <b>Production</b> or <b>Development</b> .
bytes_received	Number	The number of bytes received on the inbound request.
bytes_sent	Number	The number of bytes sent on the outbound request.
client_id	String	The unique ID of the client that is attached to the API request.
client_geoip.area_code	Number	The public switched telephone network (PSTN) area code of the client, as identified from its IP address.

Field name	Type	Description
client_geoop.city_name	String	The city name of the client, as identified from its IP address.
client_geoop.continent_code	String	The two-letter continent code of the client, as identified from its IP address.
client_geoop.country_code2	String	The two-letter country code of the client, as identified from its IP address.
client_geoop.country_code3	String	The three-letter country code of the client, as identified from its IP address.
client_geoop.country_name	String	The country name of the client, as identified from its IP address.
client_geoop.dma_code	Number	The Designated Market Area (DMA) code of the client, as identified from its IP address.
client_geoop.ip	String	The IP address of the client.
client_geoop.latitude	Number	The latitude of the client location, as identified from its IP address.
client_geoop.location	String	The longitude and latitude of the client location (separated by a comma), as identified from its IP address.
client_geoop.longitude	Number	The longitude of the client location, as identified from its IP address.
client_geoop.postal_code	String	The postal code of the client, as identified from its IP address.
client_geoop.real_region_name	String	The full name of the region that corresponds to the IP address of the client.
client_geoop.region_name	String	The abbreviated form of the region that corresponds to the IP address of the client.
client_geoop.timezone	String	The time zone of the client, as identified from its IP address.
custom_data	Array map	Custom data can be added to this field.
datetime	Date	A time stamp that records when the API was executed. The time stamp is always shown in coordinated universal time UTC.
developer_org_id	String	The identifier for the consumer organization that owns the application.
endpoint_url	String	When the request failed, identifies the proxy or invoke target URL on which the request failed. It is not included with a successful request.
developer_org_name	String	The name of the consumer organization that owns the application.
env_id	String	The Catalog identifier.
env_name	String	The name of the Catalog.
gateway_geoop.area_code	Number	The public switched telephone network (PSTN) area code of the gateway, as identified from its IP address.
gateway_geoop.city_name	String	The city name of the gateway, as identified from its IP address.
gateway_geoop.continent_code	String	The two-letter continent code of the gateway, as identified from its IP address.
gateway_geoop.country_code2	String	The two-letter country code of the gateway, as identified from its IP address.
gateway_geoop.country_code3	String	The three-letter country code of the gateway, as identified from its IP address.
gateway_geoop.country_name	String	The country name of the gateway, as identified from its IP address.
gateway_geoop.dma_code	Number	The Designated Market Area (DMA) code of the gateway, as identified from its IP address.
gateway_geoop.ip	String	The IP address of the gateway.
gateway_geoop.latitude	Number	The latitude of the gateway location, as identified from its IP address.
gateway_geoop.location	String	The longitude and latitude of the gateway location (separated by a comma), as identified from its IP address.
gateway_geoop.longitude	Number	The longitude of the gateway location, as identified from its IP address.
gateway_geoop.postal_code	String	The postal code of the gateway, as identified from its IP address.
gateway_geoop.real_region_name	String	The full name of the region that corresponds to the IP address of the gateway.
gateway_geoop.region_name	String	The abbreviated form of the region that corresponds to the IP address of the gateway.
gateway_geoop.timezone	String	The time zone of the gateway, as identified from its IP address.
gateway_ip	String	The IP address of the gateway.
global_transaction_id	String	The DataPower global transaction ID. See <a href="https://www.ibm.com/docs/en/datapower-gateway/latest?topic=variables-varserviceglobal-transaction-id-servicevarsglobaltransactionid">https://www.ibm.com/docs/en/datapower-gateway/latest?topic=variables-varserviceglobal-transaction-id-servicevarsglobaltransactionid</a> .

Field name	Type	Description
graphql_request_field_cost	Number	GraphQL APIs only. The maximum cost of all fields accessed in the query. The cost of each field access is configured in the schema.
graphql_request_type_cost	Number	GraphQL APIs only. The maximum cost of all types retrieved in the query. The cost of each type is configured in the schema.
graphql_request_max_nesting	Number	GraphQL APIs only. The maximum nesting depth found in the query by the assembly validate action. The schema configuration is used to determine which types are nested, so this value might be less than the nesting depth found by the assembly parse action.
graphql_response_field_cost	Number	GraphQL APIs only. The cost of all fields accessed in the query. The cost of each field access is configured in the schema.
graphql_response_type_cost	Number	GraphQL APIs only. The cost of all types that were retrieved in the query. The cost of each type is configured in the schema.
graphql_response_max_nesting	Number	GraphQL APIs only. The nesting depth found in the query by the assembly validate action. The schema configuration is used to determine which types are considered nested, so this value might be less than the nesting depth found by the assembly parse action.
graphql_request_top_field_counts	Object	GraphQL APIs only. The maximum number of times that each field can be retrieved by the given query. This number is equal to the number of times that the resolver is required to run. This field is stored in JSON format and is not indexed, so it is not available for visualizations. A limited number of query requests and responses are stored for each entry, based on the amount of data that each contains. The maximum amount of data that can be stored is subject to change.
graphql_request_top_type_counts	Object	GraphQL APIs only. The maximum number of times that an object of each type can be retrieved by the given query. This field is stored in JSON format and is not indexed, so it is not available for visualizations. A limited number of query requests and responses are stored for each entry, based on the amount of data that each contains. The maximum amount of data that can be stored is subject to change.
graphql_response_top_field_counts	Object	GraphQL APIs only. The number of times that each field was retrieved by the given query. This number is equal to the number of times that the resolver is required to run. This field is stored in JSON format and is not indexed, so it is not available for visualizations. A limited number of query requests and responses are stored for each entry, based on the amount of data that each contains. The maximum amount of data that can be stored is subject to change.
graphql_response_top_type_counts	Object	GraphQL APIs only. The number of times that an object of each type was retrieved by the given query. This field is stored in JSON format and is not indexed, so it is not available for visualizations. A limited number of query requests and responses are stored for each entry, based on the amount of data that each contains. The maximum amount of data that can be stored is subject to change.
headers.field_name	String	Internal information related to analytics ingestion. This is not related to the API call nor its response, see <code>request_http_headers</code> for that information.
host	String	The host name or IP address.
http_user_agent	String	The value of the User Agent header on the inbound request.
immediate_client_ip	String	The client IP address that is directly in front of the gateway. In most cases this is a load balancer.
latency_info.started	Number	The time delay (in milliseconds) between when the request was received and when the corresponding task was started by the gateway. Starting a task comprises multiple steps to prepare for executing an API; for example, completing the TCP/TLS handshake, verifying an app's client ID and secret, and matching the request URI to a Catalog, API, and Plan. When the gateway receives a request, the "Start" duration is set to 0. The duration of each step within the Start task is then added up, and the total represents the duration of the Start task.
latency_info.task	String	The API transaction that was processed.
log_policy	String	The defined logging policy. Values include none, event, headers, and payload.
org_id	String	The identifier for the provider organization that owns the API and associated Products.
org_name	String	The name of the provider organization that owns the API and associated Products.
plan_id	String	The Plan identifier.
plan_name	String	The name of the Plan. Note: The data indicates <b>N/A</b> when a client ID is not used or is invalid on the API. The gateway needs a client id to determine which app was invoking the API. From this app, the gateway can determine what plan the app is subscribed to on the product that contains the API. Without a client ID, the gateway is unable to determine what plan, product, or app was invoked because a single API can belong to multiple products (each of which have multiple plans and apps subscribed to those plans with client IDs).
plan_version	String	The version number of the Plan.
product_name	String	The Product name. Note: The data indicates <b>N/A</b> when a client ID is not used or is invalid on the API. The gateway needs a client id to determine which app was invoking the API. From this app, the gateway can determine what plan the app is subscribed to on the product that contains the API. Without a client ID, the gateway is unable to determine what plan, product, or app was invoked because a single API can belong to multiple products (each of which have multiple plans and apps subscribed to those plans with client IDs).
product_title	String	The title of the Product.
product_version	String	The version number of the Product.
query_string	String	The URL query string value on the inbound request.
rate_limit.count	Number	The number of API calls made in the defined rate limit time window.
rate_limit.limit	Number	The maximum number of requests an application is allowed to make to the API during a specified time window.
rate_limit.period	Number	The time window that is used to set a rate limit for API calls.
rate_limit.reject	String	An indication of whether calls that exceed the specified rate limit will be rejected. If true, the API call will be rejected with a 429 status code. If false, a record is created in the Activity log.
rate_limit.shared	String	An indication of whether the rate limit is shared at a Plan level by all operations, or whether a rate limit is specified on individual operations.
request_body	String	The body of the inbound request.

Field name	Type	Description
request_http_headers.field_name	String	A component of the HTTP header section of the inbound request; for example, the acceptable encodings, the identification string for the user agent, or the proxies through which the request was sent. Note: The following types of headers are considered sensitive and will not show in analytics data for security reasons: <ul style="list-style-type: none"> <li>Any secret key configured in the API security</li> <li>Any header that contains <b>secret</b></li> <li>Any header that contains <b>Authorization</b></li> </ul>
request_method	String	The method of the inbound request.
request_protocol	String	The protocol of the inbound request.
resource	String	The name of the operation.
resource_id	String	The operation identifier.
resource_path	String	The operation path.
response_body	String	The body of the outbound response.
response_http_headers.field_name	String	A component of the HTTP header section of the outbound response; for example, the MIME type of the content or the data and time when the message was sent.
status_code	String	The status code set on the outbound response.
time_to_serve_request	Number	The time elapsed (in milliseconds) from when the request was received by the backend application to when it sent a response.
transaction_id	String	The identifier for the API transaction. See <a href="https://www.ibm.com/docs/en/datapower-gateway/latest?topic=variables-varservicetransaction-id-servicevarstransactionid">https://www.ibm.com/docs/en/datapower-gateway/latest?topic=variables-varservicetransaction-id-servicevarstransactionid</a> .
uri_path	String	The URI path on the inbound request.

## Example: Event record with invocation details (activity logging setting)

```
{
 "datetime": "2016-09-29T22:17:43.404Z",
 "latency_info": [
 {
 "task": "Start",
 "started": 2
 },
 {
 "task": "security-appID",
 "started": 7
 },
 {
 "task": "Plan Limit",
 "started": 11
 },
 {
 "task": "proxy",
 "started": 12
 }
],
 "api_version": "1.0.0",
 "product_version": "1.0.0",
 "product_name": " _INTERNAL_QS_ ",
 "plan_version": "1.0.0",
 "uri_path": "/macs-shack/sb/AccountService",
 "request_method": "POST",
 "log_policy": "activity",
 "request_protocol": "https",
 "query_string": [],
 "request_body": "",
 "response_body": "",
 "bytes_received": 256,
 "bytes_sent": 256,
 "time_to_serve_request": 301,
 "status_code": "200 OK",
 "request_http_headers": [],
 "response_http_headers": [],
 "org_name": "macs-shack",
 "api_name": "accountservice",
 "catalog_name": "sb",
 "resource_path": "post",
 "plan_name": "default",
 "developer_org_name": "macs-shack"

 "client_geoip": {
 "ip": "9.20.152.215",
 "country_code2": "US",
 "country_code3": "USA",
 "country_name": "United States",
 "continent_code": "NA",
 "region_name": "NC",
 "city_name": "Durham",
 "postal_code": "27709",
 "latitude": 35.994,
 "longitude": -78.8986,
 "dma_code": 560,
 "area_code": 919,
 }
}
```



```

"timezone": "America/New York",
"real_region_name": "North Carolina",
"location": [
 -78.8986,
 35.994
],
},
"gateway_geoip": {
"ip": "9.79.12.126",
"country_code2": "US",
"country_code3": "USA",
"country_name": "United States",
"continent_code": "NA",
"region_name": "NC",
"city_name": "Durham",
"postal_code": "27709",
"latitude": 35.994,
"longitude": -78.8986,
"dma_code": 560,
"area_code": 919,
"timezone": "America/New York",
"real_region_name": "North Carolina",
"location": [
 -78.8986,
 35.994
]
}
}

```

## Example: Event record with invocation and header details (header logging setting)

```

{
"datetime": "2016-09-29T22:53:46.766Z",
"latency_info": [
 {
 "task": "Start",
 "started": 3
 },
 {
 "task": "security-appID",
 "started": 8
 },
 {
 "task": "Plan Limit",
 "started": 84
 },
 {
 "task": "activity-log",
 "started": 86
 },
 {
 "task": "proxy",
 "started": 88
 }
],
"api_version": "1.0.0",
"product_version": "1.0.0",
"product_name": "INTERNAL_QS",
"plan_version": "1.0.0",
"uri_path": "/macs-shack/sb/AccountService",
"request_method": "POST",
"log_policy": "header",
"request_protocol": "https",
"query_string": [],
"request_body": "",
"response_body": "",
"bytes_received": 256,
"bytes_sent": 256,
"time_to_serve_request": 317,
"status_code": "200 OK",
"request_http_headers": [
 {
 "Host": "apimanager.host.com"
 },
 {
 "User-Agent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:45.0) Gecko/20100101 Firefox/45.0"
 },
 {
 "Accept": "application/xml"
 },
 {
 "Accept-Language": "en-US,en;q=0.5"
 },
 {
 "Accept-Encoding": "gzip, deflate"
 },
 {
 "Content-Type": "text/xml"
 },
 {
 "SOAPAction": "getBalance"
 },
 {
 "Referer": "https://apimanager.host.com/apim/"
 }
]
}

```

```

 },
 {
 "Content-Length": "256"
 },
 {
 "Origin": "https://apimanager.host.com"
 },
 {
 "Via": "1.1 AwAABaygGU-"
 },
 {
 "X-Client-IP": "9.79.12.126"
 },
 {
 "X-Global-Transaction-ID": "1364721"
 }
],
 "response_http_headers": [
 {
 "Content-Type": "text/xml; charset=ISO-8859-1"
 },
 {
 "Date": "Thu, 29 Sep 2016 22:53:46 GMT"
 },
 {
 "X-Powered-By": "Servlet/3.0"
 },
 {
 "X-Vcap-Request-Id": "452d95be-0304-4f73-7429-7186ca6be843"
 },
 {
 "X-Global-Transaction-ID": "1364721"
 },
 {
 "Access-Control-Expose-Headers": "X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Reset, X-Global-Transaction-ID"
 },
 {
 "Access-Control-Allow-Origin": "https://apimanager.host.com"
 },
 {
 "Access-Control-Allow-Methods": "POST"
 },
 {
 "Access-Control-Allow-Credentials": "true"
 }
],
 "org_name": "macs-shack",
 "api_name": "accounts-service",
 "catalog_name": "sb",
 "resource_path": "post",
 "plan_name": "default",
 "developer_org_name": "macs-shack",

 "client_geopip": {
 "ip": "9.20.152.215",
 "country_code2": "US",
 "country_code3": "USA",
 "country_name": "United States",
 "continent_code": "NA",
 "region_name": "NC",
 "city_name": "Durham",
 "postal_code": "27709",
 "latitude": 35.994,
 "longitude": -78.8986,
 "dma_code": 560,
 "area_code": 919,
 "timezone": "America/New_York",
 "real_region_name": "North Carolina",
 "location": [
 -78.8986,
 35.994
]
 },
 "gateway_geopip": {
 "ip": "9.79.12.126",
 "country_code2": "US",
 "country_code3": "USA",
 "country_name": "United States",
 "continent_code": "NA",
 "region_name": "NC",
 "city_name": "Durham",
 "postal_code": "27709",
 "latitude": 35.994,
 "longitude": -78.8986,
 "dma_code": 560,
 "area_code": 919,
 "timezone": "America/New_York",
 "real_region_name": "North Carolina",
 "location": [
 -78.8986,
 35.994
]
 }
}

```

## Example: Event record with invocation, header, and payload details (payload logging setting)

```
{
 "datetime": "2016-09-29T22:26:28.667Z",
 "latency_info": [
 {
 "task": "Start",
 "started": 3
 },
 {
 "task": "security-appID",
 "started": 8
 },
 {
 "task": "Plan Limit",
 "started": 11
 },
 {
 "task": "activity-log",
 "started": 12
 },
 {
 "task": "proxy",
 "started": 269
 }
],
 "api_version": "1.0.0",
 "product_version": "1.0.0",
 "product_name": "_INTERNAL_QS_",
 "plan_version": "1.0.0",
 "uri_path": "/macs-shack/sb/AccountService",
 "request_method": "POST",
 "log_policy": "payload",
 "request_protocol": "https",
 "query_string": [],
 "request_body": "<SOAP-ENV:Envelope xmlns:SOAP-ENV=\"http://schemas.xmlsoap.org/soap/envelope/\"><SOAP-ENV:Header/><SOAP-ENV:Body><ban:getBalance xmlns:ban=\"http://bankA.sample.ibm.com/\"><arg0>3</arg0></ban:getBalance></SOAP-ENV:Body></SOAP-ENV:Envelope>",
 "response_body": "<soap:Envelope xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\"><soap:Body><ns2:getBalanceResponse xmlns:ns2=\"http://bankA.sample.ibm.com/\"><return>4</return></ns2:getBalanceResponse></soap:Body></soap:Envelope>",
 "bytes_received": 256,
 "bytes_sent": 256,
 "time_to_serve_request": 603,
 "status_code": "200 OK",
 "request_http_headers": [
 {
 "Host": "apimanager.host.com"
 },
 {
 "User-Agent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:45.0) Gecko/20100101 Firefox/45.0"
 },
 {
 "Accept": "application/xml"
 },
 {
 "Accept-Language": "en-US,en;q=0.5"
 },
 {
 "Accept-Encoding": "gzip, deflate"
 },
 {
 "Content-Type": "text/xml"
 },
 {
 "SOAPAction": "getBalance"
 },
 {
 "Referer": "https://apimanager.host.com/apim/"
 },
 {
 "Content-Length": "256"
 },
 {
 "Origin": "https://apimanager.host.com"
 },
 {
 "Via": "1.1 AQAAPSLVfg-"
 },
 {
 "X-Client-IP": "9.79.12.126"
 },
 {
 "X-Global-Transaction-ID": "1204915"
 }
],
 "response_http_headers": [
 {
 "Content-Type": "text/xml; charset=ISO-8859-1"
 },
 {
 "Date": "Thu, 29 Sep 2016 22:26:28 GMT"
 },
 {
 "X-Powered-By": "Servlet/3.0"
 }
]
}
```

```

 },
 {
 "X-Global-Transaction-ID": "1204915"
 },
 {
 "Access-Control-Expose-Headers": "X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Reset, X-Global-Transaction-ID"
 },
 {
 "Access-Control-Allow-Origin": "https://apimanager.host.com"
 },
 {
 "Access-Control-Allow-Methods": "POST"
 },
 {
 "Access-Control-Allow-Credentials": "true"
 }
],
 "org_name": "macs-shack",
 "api_name": "accountservice",
 "catalog_name": "sb",
 "resource_path": "post",
 "plan_name": "default",
 "developer_org_name": "macs-shack"
}

"client_geoup": {
 "ip": "9.20.152.215",
 "country_code2": "US",
 "country_code3": "USA",
 "country_name": "United States",
 "continent_code": "NA",
 "region_name": "NC",
 "city_name": "Durham",
 "postal_code": "27709",
 "latitude": 35.994,
 "longitude": -78.8986,
 "dma_code": 560,
 "area_code": 919,
 "timezone": "America/New_York",
 "real_region_name": "North Carolina",
 "location": [
 -78.8986,
 35.994
]
},
"gateway_geoup": {
 "ip": "9.79.12.126",
 "country_code2": "US",
 "country_code3": "USA",
 "country_name": "United States",
 "continent_code": "NA",
 "region_name": "NC",
 "city_name": "Durham",
 "postal_code": "27709",
 "latitude": 35.994,
 "longitude": -78.8986,
 "dma_code": 560,
 "area_code": 919,
 "timezone": "America/New_York",
 "real_region_name": "North Carolina",
 "location": [
 -78.8986,
 35.994
]
}
}
}

```

---

## Accessing analytics data with the toolkit CLI

You can access your analytics event data by using the API Connect toolkit CLI.

### About this task

The toolkit CLI provides commands for viewing your analytics data, and also for configuring analytics storage settings. To view organization scoped event data, login to the toolkit CLI using provider organization user credentials. To view cloud scoped event data and analytics storage settings, login to the toolkit CLI as the cloud admin user.

### Procedure

1. Install the toolkit: [Setting up the API Connect toolkit](#)
2. Log in to the Management server: [Toolkit CLI Login](#).

```
apic login --server <management_hostname/api_base_url> --username <username> --realm <realm>
```

Where:

- **<management\_hostname/api\_base\_url>** is the FQDN or base URL of your management API endpoints, for example: `mgmt.example.com` or `https://mgmt.example.com`
- **<username>** is either the admin user or provider org owner, depending on whether you want to access cloud or provider organization scoped analytics data.

- `<realm>` is the user registry realm for the specified `<username>`. By default this is `admin/default-idp-1` for the admin user, and `provider/default-idp-2` for provider organization users.

3. View the available toolkit analytics commands by using `-m analytics -h`. Example:

```
apic -m analytics -h
...
Available Commands:

 Creating applications and artifacts
 config Manage configuration variables

 Publishing to the cloud
 clustermgmt Clustermgmt operations
 dashboards Dashboards operations
 events Events operations
 filterparams Filterparams operations

 Other commands
 licenses Review the license for API Connect
 validate Validate an API or product definition
 version Get the APIConnect toolkit version
...
```

4. View your analytics events by using the `events` commands:

```
apic -m analytics events -h
Type "apic [command] --help" for help on the following related commands:
events:catalogCount Count events at catalog scope.
events:catalogGet Get specific event at catalog scope.
events:catalogList List events at catalog scope.
events:cloudCount Count events at cloud scope.
events:cloudGet Get specific event at cloud scope.
events:cloudList List events at cloud scope.
events:orgCount Count events at organization scope.
events:orgGet Get specific event at organization scope.
events:orgList List events at organization scope.
events:spaceCount Count events at space scope.
events:spaceGet Get specific event at space scope.
events:spaceList List events at space scope.

To get a count of all API events at cloud level:
apic -m analytics events:cloudCount --analytics-service <analytics service name> --server <management server api endpoint>
--format=json
{
 "total": 200
}

To get all the cloud level events:
apic -m analytics events:cloudList --analytics-service <analytics service name> --server <management server api endpoint>
--format=json
{
 "total": 200,
 "search_time": 7,
 "events": [
 {
 "log_policy": "activity",
 ...
 }
]
}
```

- [Example toolkit CLI operations](#)  
Examples of useful analytics CLI commands.

## Example toolkit CLI operations

Examples of useful analytics CLI commands.

The CLI help lists all the filters that can be used when querying analytics data.

```
apic-slim -m analytics events:orgList -h
Usage: apic events:orgList [flags]
```

List events at provider organization scope.

Return a result set of analytics API events scoped to the provider organization. The items in the result set are determined by the passed in request query to this API. The order of the result set is sorted from newest to oldest events by specified 'datetime'. In this API, references to 'field' refer to the analytics API event fields documented here: [https://www.ibm.com/docs/en/api-connect/10.0.5.x\\_lts?topic=data-api-event-record-fields](https://www.ibm.com/docs/en/api-connect/10.0.5.x_lts?topic=data-api-event-record-fields)

```
Flags:
--accept-language string Natural language and locale that the client prefers.
--analytics-service string Analytics service name or ID. (required)
--api_id string Filter results by API ID.
--api_name string Filter results by API name.
--app_id string Filter results by application ID.
--app_lifecycle_state string Filter results by application lifecycle state.
--app_name string Filter results by application name.
--bytes_received string Filter results by the number of bytes received.
--bytes_sent string Filter results by the number of bytes sent.
--catalog_id string Filter results by catalog ID.
--catalog_name string Filter results by catalog name.
--client_id string Filter results by client ID.
--client_ip client_ip=9.123.234.0/24 Filter results by client IP address. Only supports equality
```

checks and no operator prefixes. Supports CIDR notation based queries, e.g. `client_ip=9.123.234.0/24`

```

--consumer_org_id developer_org_id Filter results by consumer organization ID (filters on the
developer_org_id data field).
--consumer_org_name developer_org_name Filter results by consumer organization name (filters on the
developer_org_name data field).
--consumer_org_title developer_org_title Filter results by consumer organization title (filters on the
developer_org_title data field).
--end string If set, only return events with a datetime equal or older than
this. The value must follow the date-time notation as defined by RFC 3339, section 5.6, for example, 2017-07-21T17:32:28Z. See
https://datatracker.ietf.org/doc/html/rfc3339#section-5.6.
--endpoint_url string Filter results by endpoint_url, note this field is only set on
failed API events.
--event_id string If set, only return this specific event ID.
--fields string A list of the event fields that you want included in each
event. You can use a comma as a separator when specifying the fields. Use this parameter if you require only a subset of the
event fields rather than the full set.

--format string Output format. One of [json yaml go-template=... go-template-
file=...], defaults to yaml.
--gateway_ip gateway_ip=9.123.234.0/24 Filter results by gateway IP address. Only supports equality
checks and no operator prefixes. Supports CIDR notation based queries, e.g. gateway_ip=9.123.234.0/24
--global_transaction_id string Filter results by global transaction ID.
--graphql_request_field_cost string Filter results by graphql request field cost value.
--graphql_request_type_cost string Filter results by graphql request type cost value.
--graphql_response_type_cost string Filter results by graphql response type cost value.
--immediate_client_ip immediate_client_ip=9.123.234.0/24 Filter results by immediate client IP address. Only supports
equality checks and no operator prefixes. Supports CIDR notation based queries, e.g. immediate_client_ip=9.123.234.0/24
--limit string If set, the number of items to return when requesting a list.
Maximum: 500
--method string Filter results by HTTP method e.g. DELETE, GET, POST.
--offset string If set, the offset to be applied when requesting a list. e.g.
return events starting from the 100th result
-o, --org string Provider organization name or ID. (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--path string If set, only return events that contain this path value.
--path_id string Filter results by path_id
--plan_id string Filter results by plan ID
--plan_name string Filter results by plan name
--plan_version string Filter results by plan version
--product_id string Filter results by product ID.
--product_name string Filter results by product name.
--product_title string Filter results by product title.
--product_version string Filter results by product version.
--query_string string Filter results by query_string.
--request_body string Filter results by request body.
--resource_id string Filter results by resource ID.
--response_body string Filter results by response body.
-s, --server string management server endpoint (required)
--space_id string Filter results by space ID.
--space_name string Filter results by space name.
--start string If set, only return events with a datetime equal to or newer
than this. The value must follow the date-time notation as defined by RFC 3339, section 5.6, for example, 2017-07-21T17:32:28Z.
See https://datatracker.ietf.org/doc/html/rfc3339#section-5.6.

--status_code string If set, only return events that match this response code e.g.
200 OK, 404 Not Found. It is also possible to use wildcard based values 10x, 20x, 30x, 40x, 50x.
--time_to_serve request string The time in milliseconds to serve the request.
--timeframe string If set, only return events within the indicated timeframe.
This parameter will be ignored if either start or end have been specified.

```

Operators can be applied to these filters to provide advanced filtering.

Note: In the examples shown the full command is

```

apic-slim -m analytics events:orgList -s <management_fqdn> --analytics-service <analytics_service_name> --org <provider_org> --
format yaml <filter>

```

For brevity, only the base command and filter are shown in the following examples.

- Select all events with consumer organization ID of `7f9b6ee4-268d-449c-adc1-e0f3bbde8dfe`.

```

apic-slim -m analytics events:orgList ... --consumer_org_id 7f9b6ee4-268d-449c-adc1-e0f3bbde8dfe

```

Note: In the API event records, `consumer_org_id` is referred to as `developer_ord_id`.

- Select all events that do not have the consumer organization ID of `7f9b6ee4-268d-449c-adc1-e0f3bbde8dfe`.

```

apic-slim -m analytics events:orgList ... --consumer_org_id not:7f9b6ee4-268d-449c-adc1-e0f3bbde8dfe

```

- Select all events that have the product name of `superproduct`.

```

apic-slim -m analytics events:orgList ... --product_name superproduct

```

- Select all events that have a product name that contains the string `super`.

```

apic-slim -m analytics events:orgList ... --product_name contains:super

```

- Select all events that have a product name that does not contain the string `super`.

```

apic-slim -m analytics events:orgList ... --product_name notcontains:super

```

- Select all events that have a product name that starts with the string `super`.

```

apic-slim -m analytics events:orgList ... --product_name startswith:super

```

- Select all events that have a product name that ends with the string `super`.

```
apic-slim -m analytics events:orgList ... --product_name endswith:super
```

- Select all events that have one of a list of product names.

```
apic-slim -m analytics events:orgList ... --product_name oneof:superproduct,normalproduct,basicproduct
```

- Select all events that do not have one of a list of product names.

```
apic-slim -m analytics events:orgList ... --product_name notoneof:superproduct,normalproduct,basicproduct
```

- Select all events where the product name matches a regex:

```
apic-slim -m analytics events:orgList ... --product_name regex:"[a-zA-Z]amlet"
```

For regex syntax and examples, see [Regex syntax](#)

Note: Regex queries are more resource intensive and so can take longer than other methods.

- Select all events where the bytes received were 57.

```
--bytes_received 57
```

- Select all events where the bytes received were more than or equal to 57.

```
--bytes_received gte:57
```

For numeric fields, the available comparison operators are: `lt:`, `lte:`, `gt:`, `gte:`. For text and numeric fields these prefixes available are `not:`, `contains:`, `notcontains:`, `oneof:`.

Note:

The regex operator uses the following syntax:

- `"\"` is the escape character, to be used before any of the other reserved characters, for them to applied literally.

```
"a\~z" matches "a-z"
```

- `"[]"` square brackets match a single character within the brackets.

```
"[01]m" matches "0m" or "1m"
```

- `"-"` denotes a range if used between two characters.

```
"[a-z]" matches any character between "a" and "z", for example "a", "b", "c", ...
```

When used at the start of the regex `"-"` matches `"-"`.

- `"^"` is a NOT operator that negates what is specified after it.

```
"[^k]" matches any character except for "k".
```

- `"+"` denotes repetition of the preceding character at least once.

```
"U$+" matches "U$", "U$$", "U$$$$", ...
```

- `"()"` parentheses treat their contents as if they were a single character, for use in combination with operators that apply to preceding characters.

```
"tr(cat)+" matches "trcat", "trcatcat", "trcatcatcat"
```

- `"{}"` curly brackets indicate the minimum and maximum number of repetitions of the preceding character. The maximum value is unlimited if omitted with the comma, and the same as the minimum if omitted without the comma.

```
"1{3,4}" matches "111" and "1111"
```

```
"2{4}" matches "2222" only.
```

```
"2{4,}" matches "2222", "22222", "222222", ...
```

- `"?"` denotes repetition of the preceding character zero or one times.

```
"ti?" matches "t" and "ti"
```

- `"."` matches any character.

```
"da." matches "daa", "dab", ...
```

- `"*"` denotes repetition of the preceding character zero or more times.

```
"cab*" matches "ca", "cab", "cabb"
```

- `"|"` denotes the OR operator. The longest pattern on either side of the `"|"` matches.

```
"Mb|Mi" matches "Mb" and "Mi"
```

- `"#"` denotes not string, not even an empty string.

- `"&"` is the AND operator.

```
"hello.+&.+rest" matches "hello-world-rest"
```

- `"~"` denotes a fuzzy logic match.

```
"hello-world~dt" matches "hello-world-test"
```

- `"@"` matches any string.

- `"<>"` angle brackets match a number range.

```
"<3-15>Mb" matches "3Mb", "4Mb", "5Mb", ...
```

# Accessing analytics data with the REST API

You can access your Analytics event data by using the API Connect REST API.

## Before you begin

API Connect REST API calls are authenticated by using a bearer token in the authorization header. Use the toolkit credentials to request a bearer token. Toolkit credentials can be found in the API Manager UI:

1. Log in to the API Manager UI.
2. From the home page, click the Download toolkit tile.
3. Download the Toolkit credentials.
4. Open the downloaded `credentials.json` file, example:

```
...
 "toolkit": {
 "endpoint": "https://mgmt.api.example.com/api",
 "client_id": "7409693f-f726-48b4-8909-7c0d26f13e81",
 "client_secret": "5feeb0be-17f8-41a4-96d6-d40f33d69ef6"
 },
 ...

```

Take note of the `toolkit.client_id`, `toolkit.client_secret`, and `toolkit.endpoint` from this file. You will use these to get your bearer token.

## About this task

The Analytics REST API is documented here: [Analytics REST API](#).

Important: The Analytics REST API was replaced in API Connect 10.0.5 and the new Analytics REST API is not backwards-compatible. If you are using a previous version of the REST API, update calls to use the new REST API.

## Procedure

1. Use your credentials to request a bearer token:

```
curl -v -k -X POST -d '{"username": "<username>", "password": "<password>", "realm": "<realm>", "client_id": "<client_id>", "client_secret": "<client_secret>", "grant_type": "password"}' -H 'Content-Type: application/json' -H 'Accept: application/json' https://<management_server_platform_api_endpoint>/token
```

Where:

- `<username>` is either the admin user or provider org owner, depending on whether you want to access cloud or provider organization scoped analytics data.
- `<password>` is the password for the specified `<username>`.
- `<realm>` is the user registry realm for the specified `<username>`. By default the realm is `admin/default-idp-1` for the admin user, and `provider/default-idp-2` for provider organization users.
- `<client_id>` is the client ID from the "toolkit" section of the `credentials.json` file, or as provided by your API Connect cloud administrator.
- `<client_secret>` is the client secret from the "toolkit" section of the `credentials.json` file, or as provided by your API Connect cloud administrator.
- `<management_server_platform_api_endpoint>` is the platform API endpoint from the "toolkit" section of the `credentials.json` file, or as provided by your API Connect cloud administrator.

The bearer token is returned in the `access_token` property:

```
{
 "access_token": "<bearer_token>",
 "token_type": "Bearer",
 "expires_in": 28800
}
```

Note the `expires_in` value, which is the number of seconds before the token expires. After expiry you need to request a new bearer token.

2. Use the returned bearer token to call the analytics REST API:

```
This call to /orgs/<provider_organization>/events requires a bearer token that was requested with provider org credentials:
curl -v -k -H 'Accept: application/json' -H 'Authorization: Bearer <bearer_token>' -X GET --url 'https://<management_server_api_endpoint>/analytics/<analytics_service>/orgs/<provider_organization>/events'
```

```
{
 "total": 300,
 "search_time": 3,
 "events": [...]
}
```

```
This call to /cloud/events requires a bearer token that was requested with admin credentials:
curl -v -k -H 'Accept: application/json' -H 'Authorization: Bearer <bearer_token>' -X GET --url 'https://<management_server_api_endpoint>/analytics/<analytics_service>/cloud/events'
```

```
{
 "total": 45543,
 "search_time": 36,
 "events": [...]
}
```

Where:

- `<management_server_api_endpoint>` is the `toolkit.endpoint`, but with the `/api` at the end replaced with `/analytics` to access the analytics APIs.
- `<analytics_service>` is the name of the Analytics service.



---

# Modifying analytics API event data

How to modify API event data before it is stored.

## Before you begin

---

Ensure you are familiar with the API Manager UI, creating REST APIs, and viewing Analytics data.

## About this task

---

The analytics subsystem stores analytics data as individual API event records. One record represents one API event that is logged by the gateway. The API event record fields are defined here: [API event record fields](#). It is possible to customize the API event records for specific APIs. You can modify the contents of existing API event fields, and add new custom fields by using a gateway script or the `set-variable` operation.

Note: If you want to make updates to all incoming API event data, rather than specific APIs, you can use filters as described here:

- [Modifying your incoming analytics data on Kubernetes and OpenShift](#).
- [Modifying your incoming analytics data on OVA](#).

## Procedure

---


1. In the API Manager UI, click the Develop APIs and products tile, select the APIs tab, click Add, select API (from REST, GraphQL or SOAP).
2. Select New OpenAPI, give the API the title of "Hello World" and click Next. Click Next again to accept the security defaults, and then click Edit API.
3. Select the OpenAPI YAML source view `</>`, replace the generated YAML with the following YAML and click Save:

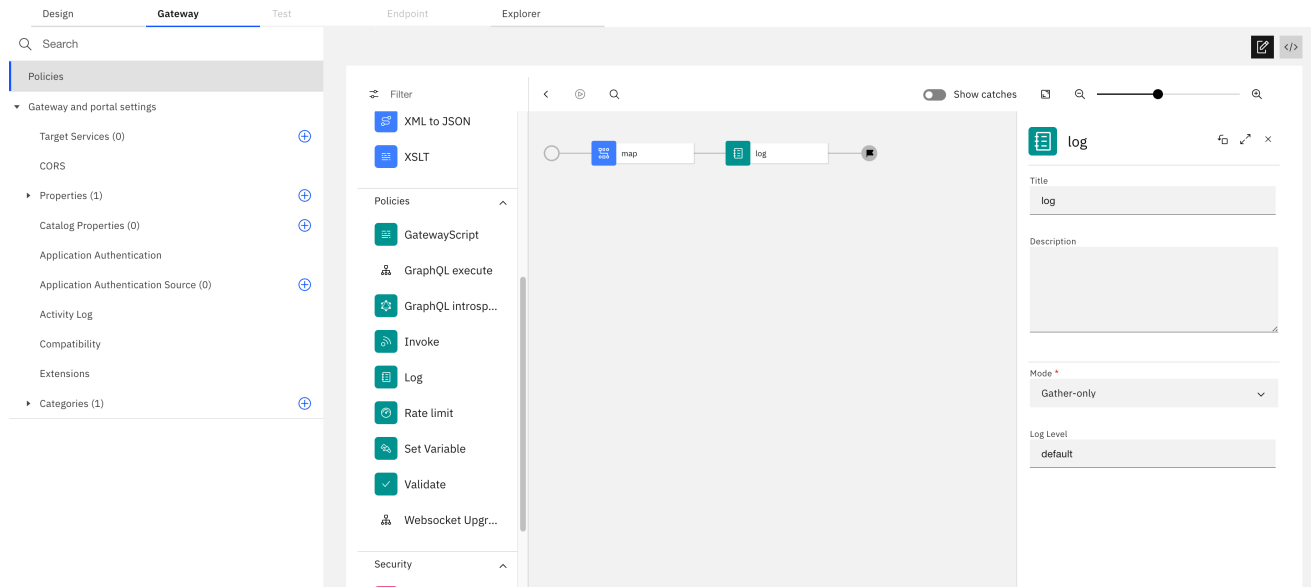
```
swagger: '2.0'
info:
 title: Hello World
 x-ibm-name: hello-world-rest
 version: 1.0.0
schemes:
 - https
basePath: /hello-world
security:
 - {}
securityDefinitions:
 clientID:
 type: apiKey
 in: header
 name: X-IBM-Client-Id
x-ibm-configuration:
 phase: realized
 testable: true
 enforced: true
 properties:
 target-url:
 value: http://example.com/operation-name
 description: The URL of the target service
 encoded: false
cors:
 enabled: true
activity-log:
 enabled: true
 error-content: header
 success-content: activity
application-authentication:
 certificate: false
assembly:
 execute:
 - map:
 version: 2.0.0
 title: map
 inputs:
 name:
 schema:
 type: string
 variable: request.parameters.name
 outputs:
 response:
 schema:
 $ref: '#/definitions/Result'
 variable: message.body
 actions:
 - set: response.hello
 from: name
 value: "'Hello ' +$(name)'"
 catch: []
 finally: []
gateway: datapower-api-gateway
categories:
 - /tests
type: rest
definitions:
 Result:
 type: object
 additionalProperties: false
 properties:
```

```

 hello:
 type: string
paths:
 /hello:
 get:
 parameters:
 - name: name
 in: query
 required: false
 type: string
 responses:
 '200':
 description: OK Response
 schema:
 $ref: '#/definitions/Result'
 parameters: []

```

4. Click the Form icon  to switch to the Form view and select the Gateway tab.
5. From the palette, select the Log policy element and drag it onto the canvas, placing it at the last position in the flow. Set the log policy to Gather-only. This step is so that the "log" context variable is populated, for use in the next step.



6. Modify API event data or add custom data either with a gateway script or by using the set-variable operation.

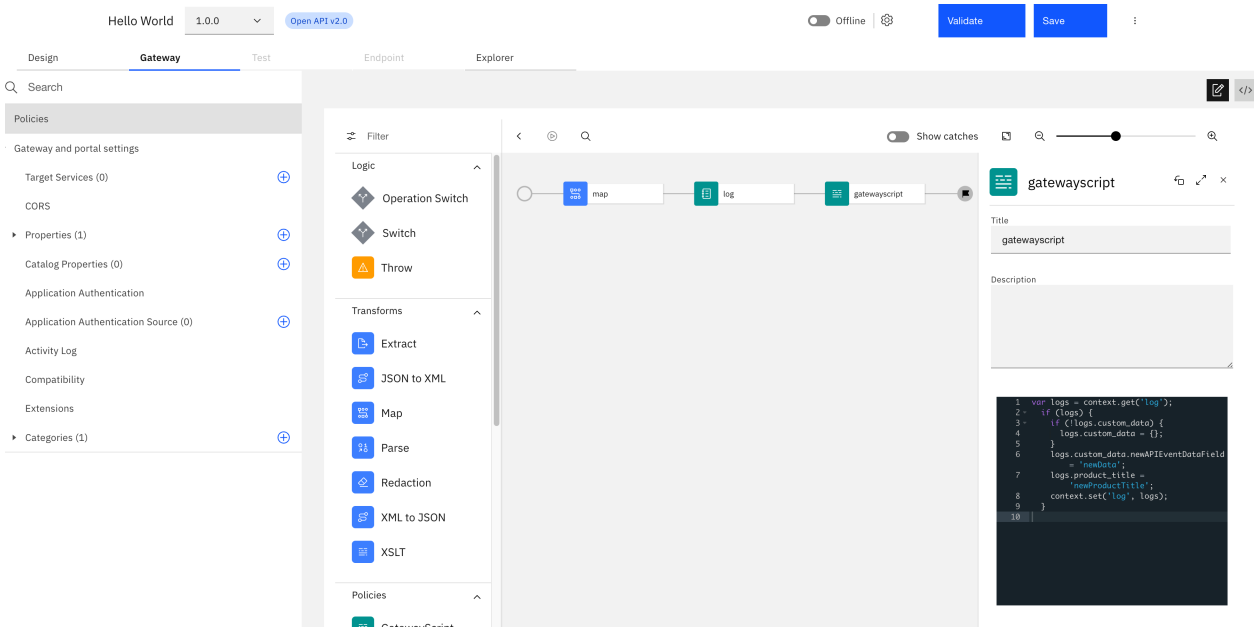
#### Gateway script method

Add a Gateway script policy to the assembly flow and paste in the following script code:

```

var logs = context.get('log');
if (logs) {
 if (!logs.custom_data) {
 logs.custom_data = {};
 }
 logs.custom_data.newAPIEventDataField = 'newData';
 logs.product_title = 'newProductTitle';
 context.set('log', logs);
}

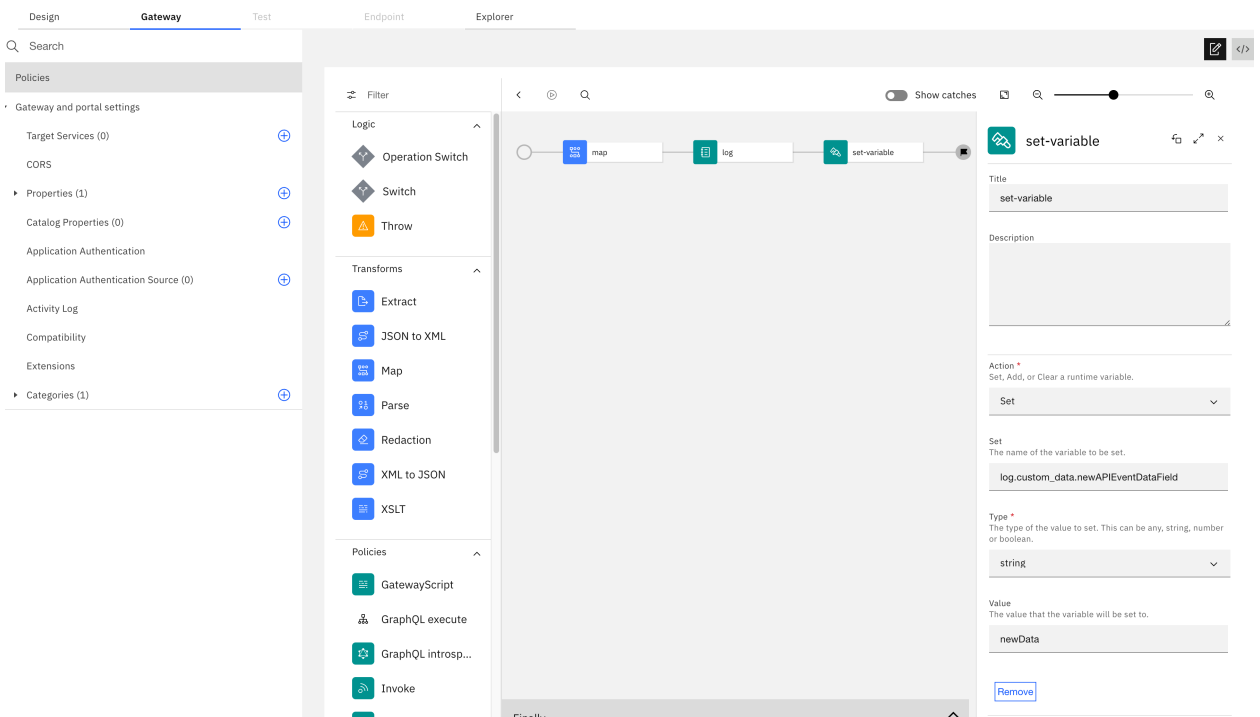
```



In this example, the new field that is added to the API event record is called **newAPIEventDataField** and its value is set to **newData**. The value of the existing field **product\_title** is changed to **newProductTitle**.

#### Set-variable method

If you cannot or do not want to use a gateway script, then you can use a set-variable policy instead. Add a set-variable policy to the assembly flow, and set it as follows:. In this example, the new field that is added to the API event record is called **newAPIEventDataField** and its value is set to **newData**.



The YAML code for set-variable is:

```

- set-variable:
 version: 2.0.0
 title: set-variable
 actions:
 - set: log.custom_data.newAPIEventDataField
 value: newData
 type: string
 - set: log.product_title
 value: newProductTitle
 type: string

```

Note: **context.log.custom\_data** supports the addition of primitive typed values such as strings and integers, and there is no enforced limit on how many you can add. Add data only to the **context.log.custom\_data** object. Do not add data to **context.log**.

The type of existing fields cannot be changed. For example, **product\_title** is a string and cannot be changed to an integer. Do not attempt to assign a different type to an API event field.

7. Publish your "Hello World" API to a catalog.

8. Make a test call to the API and verify that the new custom data is set.

a. From the Analytics view, select Discover and click the API event record corresponding to your test call:

b. Scroll down the page to view the API event record payload. Your custom data field should be present:

**Transaction ID: 1699**  
**Datetime: 03/02/2023, 11:19:09 UTC**

```
"org_id": "b9f96e8c-ec64-498b-bee8-1714e92a017b",
"gateway_ip": "9.46.120.137",
"client_ip": "9.171.72.199",
"host": "172.16.49.215",
"developer_org_id": "",
"custom_data": {
 "newAPIEventDataField": "newData"
},
"request_protocol": "https",
"domain_name": "apiconnect",
"path_id": "::GET:/hello",
"tags": [
 "apicapievent"
],
"event_id": "6oIAF4YBHjwkR4tDdR0e"
```

9. You can use filters to view only analytics data that has your new custom field set.

The screenshot shows the 'New' filter configuration page in IBM API Connect. It includes sections for 'Time range' (with radio buttons for various durations, 'Last 30 days' selected), 'Fields' (with a table for field filtering), and 'Type' (with radio buttons for event types, 'All events' selected). The 'Fields' section contains a table with the following data:

Field	Operator	Value	Delete
Custom data			
newAPIEventDataField	equals	newData	

Buttons for 'Load query', 'Add', 'Cancel', 'Save as...', and 'Apply' are also visible.

10. To use new data with REST API calls, use the custom\_data query parameter:

```
/cloud/events?custom_data[newAPIEventDataField]=newData
```

## Administering user access

If you have permission to administer users in IBM® API Connect, you can add and delete users. After users are removed from an organization through deletion, the user account remains in API Manager.

## About this task

You can also authorize users to perform different roles within your organization. To make any changes to the users that can access your organizations, use the following tasks:

- [Setting the expiration for invitations](#)  
Set a time-out for invitations to expire.
- [Configuring notifications](#)  
Specify the sender name and email address to use for email notifications. You can preview the templates and customize the text if required. You can also style the notifications to match your business theme.
- [Viewing your user information](#)  
You can view the users that can access your IBM API Connect organization.
- [Adding provider organization users and assigning roles](#)  
If you have the permissions that are required to edit users in API Connect, you can add users to your provider organization, remove users, assign roles and perform other user administration tasks.
- [Creating custom roles](#)  
If you have permission to edit roles in IBM API Connect, you can create custom roles, and assign permissions, in a provider organization. You can create as many custom roles as you want.
- [Removing a user from a provider organization](#)  
If you have permission to edit users, you can remove a user from a provider organization.
- [Configuring LDAP group mappings on API Manager user roles](#)  
As an API Manager administrator, you can configure LDAP group mapping on API Connect roles in a provider organization by using the developer toolkit CLI.

---

## Setting the expiration for invitations

Set a time-out for invitations to expire.


### About this task

---

Invitations contain a link, which expires after the specified the timeout.

### Procedure

---

1. In API Manager, click  Settings.
2. In the Settings navigation list, click Onboarding, then click Edit.
3. To specify a timeout value, select or type an integer value in the Number field and then choose a unit of time (Seconds, Minutes, or Hours) in the Unit field.
4. Click Save.

---

## Configuring notifications

Specify the sender name and email address to use for email notifications. You can preview the templates and customize the text if required. You can also style the notifications to match your business theme.

### About this task

---

The API Manager emails are sent automatically when certain system events occur. The email templates are organized by scope, as explained in Table 1.

Table 1. Notification scopes

Scope	Events that trigger the notification
catalog	Activities related to Catalogs; such as application life cycle events, invitations to catalogs, product approvals, and subscription approvals.
consumer	Activities to consumer applications and subscription requests in the Developer Portal, invitations to Consumer organizations, and password reset requests for Developer Portal accounts.
provider	Invitations.
space	Invitations.

Email notifications contain a sender name and address that's based on a hierarchical search of the configured sender details in API Connect. Depending on the email template being sent, the search can start at the Space level, and then go through the Catalog, Provider Organization, and the Cloud Manager levels. The sender details that are used will be the first set of configured details found during this hierarchical search. To view which search path API Connect takes when looking for the sender details for each notification template, see [Configuring sender details for email notifications](#).


One of the following roles is required to configure email notifications:

- Administrator
- Owner
- A custom role with the **Settings:Manage** permission

Note: You can also customize notifications by using the developer toolkit CLI, or by using the API Connect REST APIs. See [API development and management commands](#), or [API Connect REST APIs](#), for more information.


---

### Procedure

1. In API Manager, click  Settings.
2. In the Settings navigation list, click Notifications, then click Edit.
3. Specify a sender by entering a Name and Email address, then click Save.

To configure sender details at the Space or Catalog levels, you must use the developer toolkit CLI. For more information, see [Configuring sender details for email notifications](#).

- Optional: To enable template customization, move the Customize notification templates slider to the On position.  
Note: You can reload all the default notifications, as configured by a cloud administrator, by disabling and re-enabling template customization. However, doing so will overwrite any customizations that have been made. For more information on customizing notifications as a cloud administrator, see [Customizing email notification templates](#).

- To preview the text for a template, select Preview from the options menu  alongside the template name.
- To edit the text for the template, either click the template name, or select Edit in the options menu alongside the template name.  
When a template is opened, an attempt is made to get the language from the browser setting, but if this isn't possible the template defaults to English.

The text includes variables; for example, `{{catalog}}`. The notification text is based on Handlebars syntax. Most variables are enclosed in double curly braces `{{ }}`, but can be enclosed in triple curly braces `{{{ }}` to disable HTML escaping, when the variable is a URL link for example. For more information on Handlebars, see <https://handlebarsjs.com/>.

To obtain the complete list of variables that are available for a particular notification template, complete the following steps:

- Log in to the management server from the command line as a member of a provider organization; for details, see [Logging in to a management server](#). You can use the same management server URL, user name, and password in the login command that you use to log in to the API Manager user interface.
- Enter the following command:

```
apic notification-templates:get template_name --server mgmt_endpoint_url --scope org --org provider_organization --subcollection template_scope --fields variables --output -
```

where:

- `template_name` is the name of the required notification template, as displayed in the Template column in the user interface.
- `template_scope` is the scope name displayed in the Scope column alongside that template.
- `provider_organization` is the value of the `name` field for your provider organization.

For example:

```
apic notification-templates:get member-invitation --server https://myserver.com --scope org --org myorg --subcollection catalog --fields variables --output -
```

The variables that are available for the template are displayed, for example:

```
variables:
- org
- catalog
- activationLink
- expiresAt
- originator
- originatorFirstName
- originatorLastName
- originatorEmail
- username
- email
- firstName
- lastName
```

The `--output -` parameter causes the command output to be written to the command line. You can specify `--output filepath` to have the output written to a .yaml file at the specified location, or omit it altogether to have a file written to the current folder.

- To view and edit the notification template in a different language, select one of the following supported languages from the View template in drop-down list:
  - Chinese (Simplified)
  - Chinese (Traditional)
  - Czech
  - Dutch
  - English (US English)
  - French
  - German
  - Italian
  - Japanese
  - Korean
  - Polish
  - Portuguese
  - Russian
  - Spanish
  - Turkish
- Edit the Subject as required.
- Select the Content type that you want to use for the template, from HTML, PlainText, or Both. The default content type is PlainText.  
An edit window for the selected content type is displayed, or both edit windows are displayed if Both is selected.
- Edit the body of the template as required.

For HTML content, only the tags and their attributes that are shown in the following table are allowed.

Table 2. List of allowed HTML tags and their attributes

HTML tag	Attribute
<code>&lt;a&gt;</code>	"class", "href", "hreflang", "style"
<code>&lt;em&gt;</code>	"class", "style"
<code>&lt;strong&gt;</code>	"class", "style"
<code>&lt;cite&gt;</code>	"class", "style"
<code>&lt;blockquote&gt;</code>	"class", "cite", "style"
<code>&lt;code&gt;</code>	"class", "style"
<code>&lt;ul&gt;</code>	"class", "type", "style"
<code>&lt;ol&gt;</code>	"class", "start", "type", "style"
<code>&lt;li&gt;</code>	"class", "style"
<code>&lt;dl&gt;</code>	"class", "style"

HTML tag	Attribute
<dt>	"class", "style"
<dd>	"class", "style"
<h1>	"class", "id", "style"
<h2>	"class", "id", "style"
<h3>	"class", "id", "style"
<h4>	"class", "id", "style"
<h5>	"class", "id", "style"
<h6>	"class", "id", "style"
<p>	"class", "style"
<div>	"class", "style"
 	"class", "style"
<span>	"class", "style"
<img>	"class", "src", "alt", "data-entity-type", "data-entity-uuid", "data-align", "data-caption", "width", "height", "style"
<table>	"class", "id", "style"
<tr>	"class", "id", "style"
<td>	"class", "id", "style"

If an HTML tag that is not allowed is used in a notification, the tag and its contents are displayed in the email as plain text.

Images can be used by adding a `<img`

`src="https://path/to/image.png"/>` tag in the template. The `src` attribute for the image must be a fully qualified web URL, and must be externally accessible so that the email recipients can access the image. It's not possible to reference local images, they must be fully qualified URLs. It's also not possible to embed or attach images or other files in the emails.

11. Click Save when done.

Note: Edits made to a template are saved only for the specific language version that is edited.

- [Configuring sender details for email notifications](#)

Understand how the hierarchical search works for the email notification sender details, and how you can configure sender details at the Space and Catalog level.

## Configuring sender details for email notifications

Understand how the hierarchical search works for the email notification sender details, and how you can configure sender details at the Space and Catalog level.

### Before you begin

One of the following roles is required to configure sender details for email notifications:

- Administrator
- Owner
- A custom role with the `Cloud settings:Manage` or `Settings:Manage` permission

For more information about the notification templates, their scope, and how to customize them, see [Configuring notifications](#) at the Provider Organization level, and [Customizing email notification templates](#) at the Cloud Manager level.

### About this task

Email notifications are sent automatically by API Connect when certain system events occur. These notifications contain a sender name and address that's based on a hierarchical search of the configured sender details in API Connect. Depending on the email template that's being sent, the search can start at the Space level, the Catalog level, the Provider Organization level, or the Cloud level. The sender details that are used are the first set of configured details that are found during the hierarchical search.

The following table shows the starting point for the sender details hierarchical search for each email template, ordered by the scope of the template, where:

- Cloud: emails are sent from the details that are configured in the `Settings`, `Notifications` section of the Cloud Manager.
- Org: the search for sender details starts at the Provider Organization level; if not available, the Cloud details are used.
- Catalog: the search for sender details starts at the Catalog level; if not available, the Org details are used, and if they're not available, the Cloud details are used.
- Space: the search for sender details starts at the Space level; if not available, the Catalog details are used, if they're not available, the Org details are used, and if they're not available, the Cloud details are used.
- Space or Catalog: in these cases the search starts at the Space level, or at the Catalog level, depending on the user that made the request, as shown in the following list:
  - When the product lifecycle change is started by a user in a Space, the search for sender details starts at the Space level.
  - When the product lifecycle change is started by a user in a Catalog, the search for sender details starts at the Catalog level.
  - When the product lifecycle change is approved by a user in a Space, the search for sender details starts at the Space level.
  - When the product lifecycle change is approved by a user in a Catalog, the search for sender details starts at the Catalog level.

Table 1. Starting point for the sender details search for each email template

Scope	Template name	Email subject	Sender details
admin	mail-server-test-connection	Test message from IBM® API Connect.	Cloud
admin	member-invitation	Invitation to an admin organization in IBM API Connect.	Cloud
admin	password-changed	Password changed for your user account in IBM API Connect.	Cloud
admin	password-reset	Password reset request for your user account in IBM API Connect.	Cloud
provider	invitation	Invitation to create an API provider organization in IBM API Connect.	Cloud
provider	member-invitation	Invitation to an API provider organization in IBM API Connect.	Org
catalog	app-lifecycle-cancelled	Request withdrawn to <code>{{action}}</code> app <code>{{appName}}</code> .	Catalog
catalog	app-lifecycle-request	Request for approval to <code>{{action}}</code> app <code>{{appName}}</code> .	Catalog

Scope	Template name	Email subject	Sender details
catalog	invitation	Invitation to create an API catalog in IBM API Connect.	Org
catalog	member-invitation	Invitation to an API catalog in IBM API Connect.	Catalog
catalog	task-consumer-onboard-request	Consumer onboarding request in the <i>{{catalog}}</i> developer portal.	Catalog
catalog	task-product-lifecycle-approved	Request approved to <i>{{action}}</i> API product in catalog <i>{{catalog}}</i> .	Space or Catalog
catalog	task-product-lifecycle-cancelled	Request withdrawn to <i>{{action}}</i> an API product in the <i>{{catalog}}</i> catalog.	Space or Catalog
catalog	task-product-lifecycle-denied	Request denied to <i>{{action}}</i> API product in catalog <i>{{catalog}}</i> .	Space or Catalog
catalog	task-product-lifecycle-pending	Request received to <i>{{action}}</i> API product in catalog <i>{{catalog}}</i> .	Space or Catalog
catalog	task-product-lifecycle-request	Request for approval to <i>{{action}}</i> an API product in the <i>{{catalog}}</i> catalog.	Space or Catalog
catalog	task-subscription-cancelled	Request withdrawn to subscribe to an API product in the <i>{{catalog}}</i> catalog.	Catalog
catalog	task-subscription-request	Request for approval to subscribe to an API product in the <i>{{catalog}}</i> catalog.	Catalog
space	invitation	Invitation to create an API space in IBM API Connect.	Catalog
space	member-invitation	Invitation to an API space in IBM API Connect.	Space
consumer	account-approved	<i>{{catalogTitle}}</i> developer portal account registration is approved.	Catalog
consumer	account-denied	<i>{{catalogTitle}}</i> developer portal account registration is denied.	Catalog
consumer	account-pending-approval	<i>{{catalogTitle}}</i> developer portal account registration pending approval.	Catalog
consumer	app-lifecycle-approved	Request approved to <i>{{action}}</i> app <i>{{appName}}</i> .	Catalog
consumer	app-lifecycle-denied	Request denied to <i>{{action}}</i> app <i>{{appName}}</i> .	Catalog
consumer	app-lifecycle-pending	Request received to <i>{{action}}</i> app <i>{{appName}}</i> .	Catalog
consumer	app-reinstated	<i>{{appName}}</i> app reinstated in the <i>{{consumerOrg}}</i> developer portal.	Catalog
consumer	app-suspended	<i>{{appName}}</i> app suspended in the <i>{{consumerOrg}}</i> developer portal.	Catalog
consumer	invitation	Invitation to create an API consumer organization in the <i>{{catalog}}</i> developer portal.	Catalog
consumer	member-invitation	Invitation to an API consumer organization in the <i>{{catalog}}</i> developer portal.	Catalog
consumer	password-changed	Password changed for your user account in <i>{{portalTitle}}</i> developer portal.	Catalog
consumer	password-reset	Password reset request for your <i>{{catalog}}</i> developer portal account.	Catalog
consumer	sign-up	<i>{{catalogTitle}}</i> developer portal account registration.	Catalog
consumer	task-subscription-approved	API subscription request approved for app <i>{{appName}}</i> .	Catalog
consumer	task-subscription-denied	API subscription request denied for app <i>{{appName}}</i> .	Catalog
consumer	task-subscription-pending	API subscription request received for app <i>{{appName}}</i> .	Catalog

The sender address details can be configured at four levels: Cloud, Provider Organization, Catalog, and Space, and you can set a different sender address for each Provider Organization, Catalog, or Space, as required. The sender address for the Cloud can be configured in the Cloud Manager UI; see [Setting up notifications](#) for details. The sender address for the Provider Organization can be configured in the API Manager UI; see [Configuring notifications](#) for details. However, the sender address for Catalogs and Spaces can be configured only by using the toolkit CLI, as explained in the following steps.

## Procedure

- Log in to the management server from the command line as a member of a provider organization; for more information, see [Logging in to a management server](#). You can use the same management server URL, username, and password in the login command that you use to log in to the API Manager user interface.
- To update the sender address details for a Space, run the following commands.

- Run the `apic space-settings:get` command to read the current Space settings and output them to a json file. For example:

```
apic space-settings:get --server my_mgt_server --org my_provider_org --catalog my_catalog --space my_space --format json
```

Where:

- `my_mgt_server` is your management server endpoint or URL.
- `my_provider_org` is the name of your Provider Organization.
- `my_catalog` is the name of your Catalog.
- `my_space` is the name of your Space.

- Make a copy of the output file that you can then use to update the settings, for example:

```
cp space-setting.json space-setting-update.json
```

- Edit the copied settings file to update the `email_sender` block. For example:

```
vi space-setting-update.json
...
"email_sender": {
 "custom": true,
 "name": "NO REPLY: space sender name",
 "address": "space_sender_email@nomail.com"
},
...
```

Where:

- `custom` must be changed from the default setting of `false` to `true`.
- the `name` line must be added to the file, and the `space sender name` details completed.
- the `address` line must be added to the file, and the `space_sender_email@nomail.com` address details completed.

Note: Ignore all of the other configuration details that are contained in this settings file.

- Run the `space-settings:update` command to update the Space settings with the new sender address details. For example:

```
apic space-settings:update --server my_mgt_server --org my_provider_org --catalog my_catalog --space my_space space-setting-update.json
```

Where `space-setting-update.json` is the name of your updated settings file.

- To update the sender address details for a Catalog, run the following commands.

- Run the `apic catalog-settings:get` command to read the current Catalog settings and output them to a json file. For example:



```
apic catalog-settings:get --server my_mgmt_server --org my_provider_org --catalog my_catalog --format json
```

Where:

- `my_mgmt_server` is your management server endpoint or URL.
- `my_provider_org` is the name of your Provider Organization.
- `my_catalog` is the name of your Catalog.

b. Make a copy of the output file that you can then use to update the settings, for example:

```
cp catalog-setting.json catalog-setting-update.json
```

c. Edit the copied settings file to update the `email_sender` block. For example:

```
vi catalog-setting-update.json
...
"email_sender": {
 "custom": true,
 "name": "NO REPLY: catalog sender name",
 "address": "catalog_sender_email@nomail.com"
},
...
```

Where:

- `custom` must be changed from the default setting of `false` to `true`.
- `name` line must be added to the file, and the `catalog sender name` details completed.
- `address` line must be added to the file, and the `catalog_sender_email@nomail.com` address details completed.

Note: Ignore all of the other configuration details that are contained in this settings file.

d. Run the `catalog-settings:update` command to update the Catalog settings with the new sender address details. For example:

```
apic catalog-settings:update --server my_mgmt_server --org my_provider_org --catalog my_catalog catalog-setting-update.json
```

Where `catalog-setting-update.json` is the name of your updated settings file.

## Results

---

You now understand how the hierarchical search works for the sender details for each email template, and can update the sender details at the Space and Catalog level as required.

## Viewing your user information


---

You can view the users that can access your IBM® API Connect organization.

### Procedure

---

To view your organization user information, complete the following steps:

In the navigation pane of the API Manager user interface, click  Members.

### What to do next

---

You can modify your user accounts; see [Adding provider organization users and assigning roles](#). You can create custom roles; see [Creating custom roles](#).

## Adding provider organization users and assigning roles

---

If you have the permissions that are required to edit users in API Connect, you can add users to your provider organization, remove users, assign roles and perform other user administration tasks.

### About this task


---


There are two ways to add a user:

- Invite member - send an invitation email with an activation link that enables the user to complete the addition operation.
- Add member - specify the user and add them as a member of the provider organization immediately.

### Procedure

---

- To send an email invitation to a new user to register as a member of the provider organization, complete the following steps:
  1. In the navigation pane of the API Manager user interface, click  Members.
  2. Click Add > Invite member.
  3. Enter the email address of the user.
  4. Select the roles that you want to assign to the user.
  5. Click Invite.
- To specify the user and add them as a member of the Admin organization immediately, complete the following steps:

1. In the navigation pane of the API Manager user interface, click  Members.
2. Click Add > Add member.
3. Select the required user registry.  
Note: For a user registry to be available in the selection list, one of the following conditions must be satisfied:
  - The visibility of the user registry is set to Public.
  - The visibility of the user registry is set to Custom, and the specified list of provider organizations includes the one to which you are adding the user.For more information, see [Setting visibility for a user registry](#).  
The remaining procedure varies according to the type of the selected user registry, as follows:
  - Local User Registry
    - Select whether the user is an Existing user or a New User.
      - For an existing user, complete the following steps:
        - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account.
        - Select the roles that you want to assign to the user, then click Add. The user is added and their membership is immediately enabled. The specified user can log in to the API Manager user interface.
      - For a new user, complete the following steps:
        - Enter a unique user name for the new user.
        - Supply an email address, name details, and a password.
        - Select the roles that you want to assign to the user, then click Add. The user account is created, and the user membership is immediately enabled. The specified user can log in to the API Manager user interface.
  - LDAP
    - Enter the name of a user that exists in the selected user registry.
    - Select the roles that you want to assign to the user, then click Add. The user is added, and the user membership is immediately enabled. The specified user can log in to the API Manager user interface with their LDAP user name.
  - Authentication URL and OIDC
    - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account.
    - Select the roles that you want to assign to the user, then click Add. The user is added, and the user membership is immediately enabled. The specified user can log in to the API Manager user interface.

## Results

---

The user is added to the list of provider organization members. If an email invitation was sent, the status is shown as `Pending` until the recipient of the email clicks the link in the email to complete the creation of their account, after which the status changes to `Enabled`.

## What to do next

---

The new user can access the API Manager user interface. The user's authorization within API Manager is defined by the roles that are assigned to them.

## Related tasks

---

- [Creating custom roles](#)

## Creating custom roles

---

If you have permission to edit roles in IBM® API Connect, you can create custom roles, and assign permissions, in a provider organization. You can create as many custom roles as you want.

## About this task

---

You create custom roles at the provider organization level; those roles are inherited by the Catalogs and Spaces in the provider organization. You can assign permissions at the provider organization, Catalog or Space level.



For a description of the permissions, and details of the default user roles and default permissions assigned to those roles, see [API Connect user roles](#).

Note: In API Manager, the Organization Owner role has full access and cannot be edited or deleted. All other roles, including custom roles, can be deleted. If you delete a role, users lose that role. If a user loses that role, their account remains in API Manager, enabling you to add a role to the user at a future date.

## Procedure

---

To create a custom role, complete the following steps:

1. In the navigation pane of the API Manager user interface, click  Settings
2. Click Roles, then click Add.  
The Create a Role page opens.
3. Enter a role Title and an optional Summary. A Name is entered automatically.  
Note: The value in the Name field is a single string that is used to identify the role in developer toolkit CLI commands. The Title is used for display.  
To view the CLI commands to manage roles, see [apic roles](#).
4. Use the check boxes to assign permissions to the new role.
5. When you are finished, click Save.
6. To delete a role, click the options icon  alongside the role that you want to delete, click Delete, then click Delete again to confirm the role deletion.  
Note:
  - You can create and delete a role only at the provider organization level. You cannot create or delete a role at the Catalog level. Nor can you create or delete a role at the Space level.

You can, however, assign Catalog-specific permissions to the role; for details, see [Creating and configuring Catalogs](#). You can also assign Space-specific permissions; for details, see [Managing user access in a Space](#); for more information on Spaces, see [Using syndication in API Connect](#).

- If you assign Product-Drafts or Api-Drafts permissions to the role, these permissions are not inherited by the role in a Catalog or Space. These permissions apply to working with draft Products and APIs in a provider organization and are not relevant in a Catalog or Space.

## Results

---

The custom role is created and assigned the permissions that you selected.

## What to do next

---

Assign the custom role to a user.

## Removing a user from a provider organization

---

If you have permission to edit users, you can remove a user from a provider organization.

## About this task

---

After you have removed the user from your IBM® API Connect provider organization, their resources are deleted and the user cannot access any of the organization's artifacts.



Note: The user account of the deleted user remains in the associated API Connect user registry. The user can still log in to the API Manager user interface but only information links are available; there is no access to view or manage any resources.

If you later invite the same user again, they can re-activate their account. If you are using a Local User Registry, the user must re-activate their account by using the Sign In option, **not** by completing the registration form and using the Sign Up option; attempting to re-register will fail. For details on how to invite a user to join your organization, see [Adding provider organization users and assigning roles](#).

## Procedure

---

To remove a user from a provider organization complete the following steps:

1. In the navigation pane of the API Manager user interface, click  Members.
2. Alongside the user that you want to delete, click the options icon , click Delete, then click Delete again to confirm the user deletion.  
Note: You cannot delete the owner of a provider organization.

## Results

---

The user is removed from the provider organization, and the user account remains in API Manager.

## Configuring LDAP group mappings on API Manager user roles

---

As an API Manager administrator, you can configure LDAP group mapping on API Connect roles in a provider organization by using the developer toolkit CLI.

## Before you begin

---

You must have an LDAP user registry resource in the API Manager that has the `external_group_mapping_enabled` configuration set to `true`. See [Using the CLI to create an organization-specific LDAP user registry](#) for information.

One of the following roles is required to edit roles:

- Administrator
- Owner
- Custom role with the `Settings: manage permissions`

Note: The provider organization roles apply only to API Manager users. If you want to apply LDAP group mapping to Cloud Manager users, see [Configuring LDAP group mapping on Cloud Manager user roles](#). LDAP group mapping cannot be applied to Developer Portal users.

## About this task

---

You can map external LDAP groups to the API Connect preconfigured user roles (except for the Owner and Member roles), as well as to any custom user roles, to reflect your business needs.

Notes:

- After LDAP group mapping is enabled on a role, user onboarding always honors the group mappings.
- Once on-boarded, user membership in API Connect is valid throughout the login period (`access_token_ttl`), irrespective of any changes in the external LDAP registry. Membership is updated only on the next login, when the LDAP information is fetched and refreshed.
- One or more API Connect roles can be mapped to one or more LDAP groups, and one or more LDAP groups can be mapped to a role.
- When multiple LDAP groups are mapped to a single role, it means that a user from any one of the LDAP groups can logon to API Connect.
- If a user is removed from the external LDAP user registry, to ensure quick removal from API Connect you must also delete the user membership in API Connect.

- In API Manager you can map roles at the provider organization, Catalog and, if appropriate, Space level. However, you can only map roles at the child level of Catalog, and then the Catalog's child level of Space, if the parent level also has group mapping defined. Mapping at a child level overrides the mapping for the same role at the parent level. For example:  
If the Administrator Role at the provider organization level is mapped to the LDAP group of `cn=APIC-Administrators,ou=ibmgroups,o=ibm.com`, and the Administrator Role at a Catalog level is mapped to the LDAP group of `cn=APIC-Developers,ou=ibmgroups,o=ibm.com`, then any user that is added as a Catalog member must belong to the `cn=APIC-Developers,ou=ibmgroups,o=ibm.com` group. Note that provider organization level members that belong to the `cn=APIC-Administrators,ou=ibmgroups,o=ibm.com` group can still access the Catalog due to inheritance.

You can configure LDAP group mappings using one of the following methods:

- [Using the UI to configure LDAP group mappings](#)
- [Using the CLI to configure LDAP group mappings](#)

## Using the UI to configure LDAP group mappings

Use the API Manager UI to configure LDAP group mappings.

### Procedure

1. Login to the API Manager as a user with the appropriate permissions.
2. Click Settings > Roles > Add a new Role or edit existing role .
3. Select the following settings:
  - Enable external group mapping - select to enable external group mapping for this role.
  - LDAP user Registry - LDAP user registry list which has external group mapping enabled. User must select one of the LDAP user registry from this list.
  - LDAP group names - add one or more LDAP group names that you want to map to the user role.
  - User group prefix - add the prefix for the LDAP user group.
  - User group suffix - add the suffix for the LDAP user group.
4. Save your changes.

### Results

The role is updated with the LDAP group mapping information. Users can now log on to the API Manager, and automatically be assigned the correct access permissions for their role.

## Using the CLI to configure LDAP group mappings

Use the toolkit CLI to configure LDAP group mappings.

### About this task

For detailed information about how to use the CLI, see [Installing the toolkit](#), and [Overview of the command-line tool](#).

### Procedure

Perform the following steps to map external LDAP groups to API Manager user roles.

1. Log in to the management server CLI.  
Before you can update the role configuration, you must log in to your management server from the developer toolkit CLI as a member of a provider organization. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

where `mgmt_endpoint_url` is the platform API endpoint URL.

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details of the login command, see [Logging in to a management server](#).

2. Run the following command to get the URL of the LDAP user registry resource in the API Manager that you want to map the user roles to:

```
apic user-registries:get ldap_user_registry --org organization_name --server mgmt_endpoint_url --output -
```

where:

- `ldap_user_registry` is the name or ID of your LDAP user registry resource.
- `organization_name` is the value of the `name` property of your provider organization.
- `mgmt_endpoint_url` is the platform API endpoint URL.

This command outputs the configuration details of your LDAP user registry, and the `url` is shown at the end of the list, for example:

```
type: user_registry
api_version: 2.0.0
```

```

id: 35e75bad-1d89-4a65-a70f-xxxxxx
name: ldap
title: LDAP
integration_url: >-
 https://server.com/api/cloud/integrations/user-registry/147b5fb1-e88e-41e3-90e9-xxxxxx
registry_type: ldap
user_managed: false
user_registry_managed: false
external_group_mapping_enabled: true
...
url: >-
 https://server.com/api/user-registries/3d58ce7e-16a8-493b-9684-xxxxxx/35e75bad-1d89-4a65-a70f-xxxxxx

```

3. Create a role yaml file that contains the following LDAP group mapping configuration properties:

```

external_group_mapping:
 user_registry_url: https://server.com/api/user-registries/3d58ce7e-16a8-493b-9684-xxxxxx/35e75bad-1d89-4a65-a70f-xxxxxx
 ldap_groups:
 - 'cn=apic-administrators,ou=ibmgroups,o=ibm.com'
 - 'cn=apic-developers,ou=ibmgroups,o=ibm.com'
 user_group_filter_prefix: (&(uniquemember=
 user_group_filter_suffix:)(objectClass=groupOfUniqueNames))

```

Where:

- `user_registry_url` is the URL of your LDAP user registry resource from Step [#task\\_apim\\_role\\_mapping\\_ldap\\_url](#).
- `ldap_groups` is a list of the LDAP group names that you want to map to the user role.
- `user_group_filter_prefix` is the prefix for the LDAP user group.
- `user_group_filter_suffix` is the suffix for the LDAP user group.

4. Run the following command to update the user role with the `external_group_mapping` configuration properties:

```
apic roles:update role_name --scope org --org organization_name --server mgmt_endpoint_url mapping_properties_file
```

Where:

- `role_name` is the name of the user role that you want to add the LDAP group mapping to.
- `--scope` is the organization level that you want the update to apply to. Valid values are:
  - `org` to apply the mapping at the provider organization level.
  - `catalog` to apply the mapping at the Catalog level.
  - `space` to apply the mapping at the Space level.
- `organization_name` is the value of the `name` property of your provider organization.
- `mgmt_endpoint_url` is the platform API endpoint URL.
- `mapping_properties_file` is the name of your mapping properties file from Step [#task\\_apim\\_role\\_mapping\\_role\\_file](#), for example `role_mapping_file.yaml`.

If you prefer to enter the configuration properties interactively on the command line, you can substitute the `mapping_properties_file` for a terminating hyphen character `-`, and enter the information manually, followed by pressing **CTRL**  
**D** to terminate the input.

If you want to create a custom role that includes LDAP group mapping, you can include the `external_group_mapping` configuration section in the `role_file`, and then create the new role by using the `apic roles:create` command.

For more information about the `apic roles` commands, see [apic roles](#) in the CLI reference section.

## Results


The role is updated with the LDAP group mapping information. Users can now log on to the API Manager, and automatically be assigned the correct access permissions for their role.

## Changing your API Manager password

You can change your API Manager password in IBM® API Connect.

### Procedure

To change your password, complete the following steps:

1. Log in to the API Manager user interface.
2. Click the User icon  then click My Account.
3. In the Change password section, enter your current password and your new password, and confirm the new password.  
Passwords must satisfy the following requirements:

- Cannot be re-used for at least 8 password cycles
- Cannot contain more than 2 consecutive repeating characters
- Must include at least 8 characters
- Must include 1 or more characters from at least 2 of the following categories:
  - Lowercase letters
  - Uppercase letters
  - Numbers
  - Special characters – only the following special characters are allowed:

```

!
\
#
$

```

```
%
&
\
(
)
*
+
/
-
.
/
:
;
<
=
>
?
@
[
\
]
^
_
{
|
}
~
```

4. Click Change Password.

## Results

Your password is changed.

## Resolving login problems by increasing HTTP header size

You can resolve login problems for the API Manager UI by increasing the maximum HTTP client header size.

### About this task

Login attempts to the user interface might fail with error 502 (**Bad Gateway**). This error can occur when logging in from a browser that has a large number of cookies. The error occurs because size of the login request exceeds the maximum HTTP client header size.

The maximum HTTP client header size is limited for security reasons. To workaround this issue, you can clear the browser cache and cookies, or open an incognito window from the browser, and then retry the login.

Alternatively, you can increase the maximum HTTP client header size. Use caution when increasing the maximum size because larger headers can raise a security risk to your system.

### Procedure

1. **SSH** to your management system or appliance.
2. Enter the following commands:

```
kubectl get deployment -n namespace | grep apim-v2
kubectl edit deployment -n namespace apiconnect-apim-v2-deployment
```

3. In the `env` section of the deployment configuration, change `--max-http-header-size=12000` to a larger value that works for your environment. For example:

```
- name: NODE_OPTIONS
 value: --max-old-space-size=8094 --max-http-header-size=16000
```

4. Save the updated configuration.
5. Run `kubectl get pod -n namespace` a few times until the new `apiconnect-apim-v2` pod is up and running.

API Gateway only

## Reviewing a gateway's processing status

Review the processing status for the Gateway services that are enabled in your API Connect Catalogs and Spaces.

### About this task


Sometimes you might want to check the status of an API event that is being processed by the gateway. For example, you might want to ensure that a Product was published before and is available on the gateway before you invoke one of its APIs. Or perhaps it appears that API events are being processed too slowly and you want to see if all of the events for a particular gateway service are affected. You can review a gateway's processing status to see the which events were sent for processing and which events are still waiting in the queue.

Notes:




- You can only obtain processing status information for the DataPower® API Gateway. This feature is not supported for the DataPower Gateway (v5 compatible) service.
- Processing information is collected for each Catalog. If you view information for a particular Space, the display shows the results for the entire Catalog instead.

## Procedure


1. Open the Gateway services page:

- a. Log in to API Manager.
- b. In the navigation pane, click  Manage.
- c. Select a Catalog.
- d. Click the Catalog settings tab.
- e. On the Settings page, click Gateway services.

A list of all of the gateway services for the selected Catalog, including all of its Spaces, displays the gateway type and its endpoint URL. In addition, a status icon displays for each service name to provide a quick indication of that service's current status:

-  Normal: Between 0 and 10 events are waiting to be processed
-  Increased: Between 10 and 20 events are waiting to be processed
-  High: More than 20 events are waiting to be processed

You can view details about a service's status to gain insight into possible issues.

2. On the Gateway services page, click  next to a service and select View gateway processing status.

The Gateway processing status page displays a list of the events that were sent to the gateway. For each event, you can see the following information:

- **Event type:** The action performed by the event; for example, subscribing to a product, deleting an application, or creating a new consumer organization.
- **Category:** The feature area affected by the event; for example, subscriptions, applications, or consumer organizations.
- **Time:** The time when the event was processed by the gateway service.
- **Status:** The current event's processing status; for example, Queued (not yet sent for processing), Sent (but not processed yet), or Processed (complete). The number of events in the Queued and Sent states determine the status level for the service.

The display includes events that were not processed yet, as well as the last event processed. If a particular event is not listed as Sent or Queued, then that event was already processed.

## Reference

Reference information for the API Manager component in IBM® API Connect.

- [API gateway response codes](#)  
When an API is called, different HTTP status codes are returned by the gateway to indicate whether the request was successfully completed.
- [Troubleshooting your billing configuration](#)  
Some issues can occur when you are setting up billing integration in IBM API Connect.

## Related information

- [IBM API Connect overview](#)

## API gateway response codes

When an API is called, different HTTP status codes are returned by the gateway to indicate whether the request was successfully completed.

The response codes used in IBM® API Connect correspond to the registered HTTP status codes that are typically generated to provide informational (1xx), successful (2xx), redirection (3xx), client error (4xx), or server error (5xx) responses, as described at <https://tools.ietf.org/html/rfc7231#section-6> and [Hypertext Transfer Protocol \(HTTP\) Status Code Registry](#).

In API Connect, successful responses vary depending on the API being called. Other response codes can also be generated, depending on the implementation of the assembly and the response from the external systems. The standard reasons listed for the registered HTTP status codes are considered adequate for most responses that are returned; these response codes and their causes are therefore not listed here.

In certain cases, a client or server error response code can be caused by a condition that is specific to API Connect. The following table contains a list of these error response codes and identifies possible causes for these codes being returned. For some error codes, multiple causes are possible.

Table 1. Error codes and their causes

Error Code	Cause
401 Unauthorized	<ul style="list-style-type: none"><li>• The required client identification has not been successfully provided.</li><li>• User authentication failed or did not take place.</li></ul>
403 Forbidden	<ul style="list-style-type: none"><li>• The application is not registered with the plan that is used.</li><li>• The application is not active.</li><li>• User authentication failed because multiple client IDs provided.</li></ul>

Error Code	Cause
404 Not Found	<ul style="list-style-type: none"> <li>Information for the provider organization or environment was not found.</li> <li>The API URL was not found in the organization or environment.</li> </ul>
405 Method Not Allowed	The API URL was found, but no operation was found that supports the requested HTTP verb.
406 Not Acceptable	The API cannot produce any responses that are supported by the application.
429 Too Many Requests	The rate limit has been exceeded for the plan or operation being used.
500 Internal Server Error	An error occurred while executing this request.
503 Service Unavailable	The status of an API was switched from online to offline, making the API unavailable across all Products in which it is contained. For more information, see <a href="#">Managing your Products in the API Manager UI</a> and <a href="#">Managing API Products using the developer toolkit</a> .

## Troubleshooting your billing configuration

Some issues can occur when you are setting up billing integration in IBM® API Connect.

You can use the following links to browse the issues:

- [My customer payment is not completing with Stripe, but the Product is still being accessed.](#)
- [My Product with billing is returning an error when I try to configure it with my Stripe account.](#)
- [My customer previously had a Stripe subscription to a Product but can't access its billing details.](#)

### My customer payment is not completing with Stripe, but the Product is still being accessed.

When payment issues occur with the credit card payment provider, the credit card subscription is canceled, but the API Connect subscription is not canceled. You can disable an application until the payment issues are resolved by completing the following steps:

- Navigate to the Settings page in the dashboard of your Stripe account.
- In the Billing section navigate to the Subscriptions and emails page.
- In the Manage failed payments section, change the Subscription status to If all retries for a payment fail, mark the subscription as unpaid.
- Navigate to the Subscriptions page, and when the subscription has moved into the unpaid state open the subscription in the Stripe dashboard.
- In the Metadata section, copy the subscription\_url value.
- Use the developer toolkit CLI, and the UUID from the subscription\_url, to set the state of the application or subscription to disabled.

### My Product with billing is returning an error when I try to configure it with my Stripe account.

The most common cause of this is that your Developer Portal cluster servers or your Management cluster servers cannot communicate with Stripe. To resolve this, ensure that the Stripe API is enabled with HTTPS communication with your Management cluster servers and the Developer Portal on port 443. See [Firewall requirements on Kubernetes](#) and [Firewall requirements on VMware](#) for more information about this requirement.

Other causes might be issues with the subscription billing service, network stability, or the API keys might have been rolled and not updated in API Connect and the Developer Portal.

### My customer previously had a Stripe subscription to a Product but can't access its billing details.

There is more than one possible cause and resolution for this situation.

- The credit card transaction might not be completing correctly. If this situation happens, the subscription might be canceled with Stripe, but the API Connect subscription is not canceled. For instructions about how to disable a subscription, see [My customer payment is not completing with Stripe, but the Product is still being accessed](#).
- A recent change to a Product with billing might have triggered an action that was not completed correctly. Seemingly minor changes to the Product can start a number of actions that can overload a server. For example, when you replace a version of the Product with a new version, it starts an update to the Stripe server for every account that is subscribed to that Product. If the action fails to complete, the Plan's subscription status might not be reflected correctly with Stripe. You can identify this situation by viewing the Billing dashboard in the Resources section of the API Manager. The table displays the state of the job queue for the billing integration resource, and any issues with the job queue are displayed here. For more information about the job queue, and how to retry blocked jobs, see [Billing integration resource job queues](#).

## Authoring policies

With the on-premise offering, you can control specific processing features in the Gateway server of IBM® API Connect by creating user-defined policies.

IBM API Connect enables organizations to easily promote business services as APIs to internal and external developer communities. API developers can rapidly create, proxy, assemble, and secure APIs through the API Designer user interface (UI). The API Designer assembly feature allows API developers to create complex REST or SOAP API operations that transform data, perform multiple service calls, aggregate data, and apply policies.

A policy is a piece of configuration that controls a specific aspect of processing in the IBM DataPower® Gateway server during the handling of an API invocation at run time. IBM API Connect provides a number of different types of policies, but you can also create user-defined policies to provide more processing control.

### When would you need a user-defined policy?

A user-defined policy enables you to control extra processing features in the Gateway server, such as security, or routing of requests. The user-defined policies feature is available only with the on-premise offering of IBM API Connect.



Create a user-defined policy in IBM API Connect when you need to augment the actions or activities that are performed by the API gateway. For example, you might want to perform the following operations:

- Implement your own proprietary logic for dynamic routing of requests.
- Enforce extra security constraints to your API.
- Make accessible an extra capability that is provided in DataPower that is not yet accessible in the IBM API Connect policy catalog.

Note: For best results, do not try to update your existing user-defined policies, because those are already in use by your APIs. Instead, create a new version of the policy that you want to update, and then modify the API assembly to use the newer version of the policy.

See the following topics for more information about user-defined policies, and how to create and import them into IBM API Connect:

- [Authoring policies for the DataPower Gateway \(v5 compatible\)](#)  
A user-defined policy for the DataPower Gateway (v5 compatible) consists of a package that contains a YAML definition file that describes the policy, together with the policy implementation files. You import the package into one or more Catalogs in your IBM API Connect environment to make the policy available to APIs that are published to those Catalogs.
- [Authoring policies for the DataPower API Gateway](#)  
A user-defined policy for the DataPower API Gateway consists of a package containing configuration details that define the actions of the policy. You publish the package to the DataPower API Gateway to make it available to APIs that are deployed there.

## Related information

- [IBM API Connect Overview](#)
- [API policies](#)

DataPower Gateway (v5 compatible) only

## Authoring policies for the DataPower Gateway (v5 compatible)

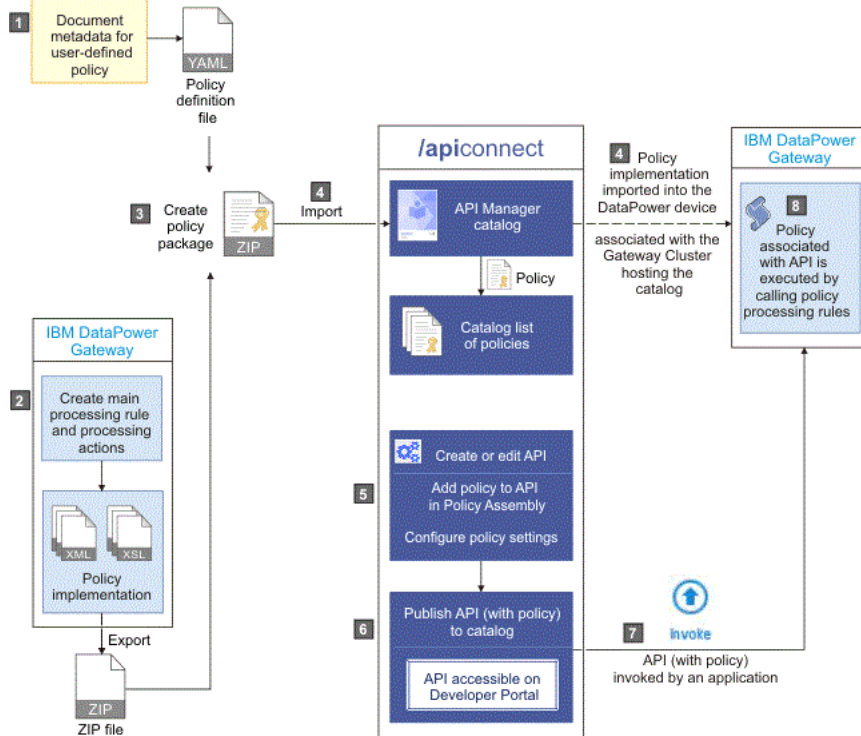
A user-defined policy for the DataPower® Gateway (v5 compatible) consists of a package that contains a YAML definition file that describes the policy, together with the policy implementation files. You import the package into one or more Catalogs in your IBM® API Connect environment to make the policy available to APIs that are published to those Catalogs.

## How do user-defined policies work?

A user-defined policy is implemented as a DataPower processing rule. After a policy is imported into an IBM API Connect Catalog, the policy is available to be placed into an assembly flow of an enforced API. When the API is published, and invoked by an application, the API Gateway executes all policies that are associated with this API. An API Gateway that is running in DataPower calls the IBM DataPower Gateway processing rules that those policies implement.

The following diagram provides an overview of how to create and execute a user-defined policy in IBM API Connect.

Figure 1. The user-defined policy process



For details on how to define and deploy user-defined policies for the DataPower Gateway (v5 compatible), see the following subtopics:

- [Describing your policy](#)  
Describe your user-defined policy by creating a definition file in YAML format to document metadata about the policy.

- [Creating a new user-defined policy](#)  
Create a user-defined policy by configuring a policy definition file and its implementation, and then packaging the policy and importing it into IBM API Connect.
- [Packaging and importing your policies into IBM API Connect](#)  
Make your user-defined policy available to API developers by packaging it and importing it into an IBM API Connect Catalog.
- [Reference](#)  
Reference information for authoring policies in IBM API Connect.

DataPower Gateway (v5 compatible) only

## Describing your policy

Describe your user-defined policy by creating a definition file in YAML format to document metadata about the policy.

### About this task

A policy YAML file contains the following sections:

- A specification version.
- An information section.
- An attach section.
- A properties section.
- A gateways section.

You can create a policy YAML file by using any editor of your choice. Both .yaml and .yml file extensions are supported, but the use of the .yaml file extension is recommended by [yaml.org](http://yaml.org).

Note: All keys and enumeration values that are specified in this topic are case-sensitive.

### Procedure

The following steps describe how to construct a policy YAML file.

1. Set the specification version by adding the following line to the beginning of the file: `policy: 1.0.0`.
2. Complete the information section with details about the policy by using the following syntax:

```
info:
 title: Title of policy
 name: shortnameofpolicy
 version: 1.0.0
 description: An example policy
 contact:
 name: name1
 url: url1
 email: email1
```

where:

- `title` is the title of the policy. Any string can be used, but the title should be kept short so that it can be displayed in the API Manager user interface.
- `name` is the short name for the policy and must contain only alphanumeric characters, the - (dash) character, and the \_ (underscore) character (blank spaces are not supported). The name is case-sensitive, and should be 20 characters or less so that it can be displayed in the API Manager user interface. In the example, the name is shown as `<policy-name>`.

Important: The policy short name must be different from the OpenAPI names of the built-in policies, otherwise it will cause a policy conflict in the API assembly definition. For a list of the OpenAPI built-in policy names, see [execute](#). Also, consider including an appropriate unique prefix or suffix string in the short names of your user-defined policies to prevent possible name conflicts with future built-in policies.

Note: The `name` property must be identical to the Node.js module name.

- `version` is the version number of the policy. This is a reserved property.  
Tip: The `version.release.modification` version numbering scheme is recommended, for example `1.0.0`.  
Note: The `info.version` in `policy.yaml` must match the `version` indicated in `package.json`.
- `description` (optional) is a short description of the policy.
- `contact` (optional) is the contact information for the policy, where:
  - `name` (string) is the identifying name of the contact person or organization.
  - `url` (string) is the URL that points to the contact information.
  - `email` (string) is the email address of the contact person or organization.

3. Complete the attach section by specifying which type of API flow the policy can be attached to.

```
attach:
 - rest
 - soap
```

where:

- `rest` indicates that this policy can be attached to a REST API flow.
- `soap` indicates that this policy can be attached to a SOAP API flow.

The attach section must contain at least one API flow type. If a policy can be attached to both API flow types, they must both be indicated in the attach section.

4. Complete the properties section by defining a JSON schema (based on JSON Schema Specification Draft 4, but rendered in YAML format) that contains the list of properties the policy will declare as required input. These properties are presented to the API author at development time, when the properties can be configured or mapped to values as required.

The JSON schema defines a root object that contains the following JSON properties:

- `type`
- `properties`
- `required`

The syntax and definition of these properties matches the JSON schema definition. The following code block shows an example of a simple schema that defines one required property (*a\_property*):

```
properties:
 $schema: "http://json-schema.org/draft-04/schema#"
 type: object
 properties:
 a_property:
 label: a label
 description: a description
 type: a type
 required:
 - a_property
```

The root *type* of the schema must be an **object**, as shown in the example schema. You can define zero or more properties in the schema. Required properties are declared by using the **required** parameter, as shown in the example schema. The policy will not be accessible in the API Manager assembly editor, unless all the required properties have values. Each property is an object with the following items:

- **label** is a short name for the property and is displayed in the Name column of the API Manager assembly editor.
  - **description** is a short description for the property and is displayed in the Description column of the API Manager assembly editor.
  - **type** must be one of the following primitives that are supported:
    - *integer*
    - *number*
    - *string*
    - *boolean* (this primitive is shown as a check box in the assembly editor)
    - *array*
  - **default** (optional) specifies a default value for the property.
  - **enum** (for properties with a **type** of *string*) is an array of valid values.
5. In your gateways section, specify that the policy is for the DataPower® Gateway. Enter the following code:

```
gateways:
 - datapower-gateway
```

6. Save the new policy as a YAML file using the value of the **name** property (in the **info** section) as the file's name. Required: To ensure a successful import, the YAML file must use the name specified in the policy's **name** property.

## Results

---

You have created a user-defined policy definition YAML file that documents metadata about your policy.

## Example

---

The following code block shows an example of a policy definition file that contains all the supported primitives. This policy must be saved as `sampleimpl.yaml` to ensure that it can be imported into API Connect.

```
policy: 1.0.0

info:
 title: Sample Policy
 name: sampleimpl
 version: 1.0.1
 description: This is a sample policy.
 contact:
 name: Steve Product Manager
 url: http://developer.acme.com/contacturl
 email: steve-product-manager@someemailservice.com

attach:
 - rest
 - soap

properties:
 $schema: "http://json-schema.org/draft-04/schema#"
 type: object
 properties:
 samplestring:
 label: String Property
 description: Sample string property
 type: string
 sampleboolean:
 label: Boolean Property
 description: Sample boolean property
 type: boolean
 sampleinteger:
 label: Integer Property
 description: Sample integer property
 type: integer
 samplenumber:
 label: Number Property
 description: Sample number property
 type: number
 samplestringwithenum:
 label: Enum Property
 description: Sample string enum property
 enum:
 - one
 - two
 - three
 default: one
```

```

 type: string
 samplearray:
 label: Array of properties
 description: Sample array of properties
 type: array
 items:
 type: object
 properties:
 string:
 label: String Property
 description: Sample string property in array
 type: string
 boolean:
 label: Boolean Property
 description: Sample boolean property in array
 type: boolean
 required:
 - string
 required:
 - samplestring
 - sampleboolean
 - sampleinteger
 - samplenumber
 - samplearray

```

```

gateways:
 - datapower-gateway

```

In the following code block, an example of a policy definition file for modifying message payload is shown.

```

policy: 1.0.0

info:
 title: Run GatewayScript
 name: gatewayscript-policy
 version: 1.0.0
 description: Execute GatewayScript
 contact:
 name: IBM DataPower Samples
 url: https://github.com/ibm-datapower/
 email: steve-product-manager@ibm.com

attach:
 - rest
 - soap

gateways:
 - datapower-gateway

properties:
 $schema: "http://json-schema.org/draft-04/schema#"
 type: object
 properties:
 source:
 label: "GatewayScript Source"
 description: "The location of the GatewayScript file to execute"
 type: string
 default: store:///identity.js
 value:
 label: "New Value"
 description: "The value to be added to the payload"
 type: string
 default: "Hello Policy"
 required:
 - source
 - value

```

## What to do next

Create an implementation for your user-defined policy by using DataPower processing rules and actions. For more information, see [Implementing your policy](#).

## Related concepts

- [Authoring policies](#)

DataPower Gateway (v5 compatible) only

## Creating a new user-defined policy

Create a user-defined policy by configuring a policy definition file and its implementation, and then packaging the policy and importing it into IBM® API Connect.

Review the following topics to learn how to create a user-defined policy.

Tip: Some sample policies are available on GitHub, and these policies can be downloaded and adapted for use with IBM API Connect. For more information, see [API Connect Policies on GitHub](#).

- [Implementing your policy](#)  
Create an implementation for your user-defined policy by using DataPower processing rules and actions.

## Related information

---

- [IBM API Connect Overview](#)

DataPower Gateway (v5 compatible) only

## Implementing your policy

---

Create an implementation for your user-defined policy by using DataPower® processing rules and actions.

### Before you begin

---

You must create a policy YAML file before you create an implementation. For more information, see [Describing your policy](#).

You can create an implementation for your policy by using the usual tools available to IBM® DataPower Gateway developers. However, the DataPower user interface is typically the tool that is used to create processing rules and actions.

Note: You must be skilled in DataPower tooling and concepts, before you can create a policy implementation.

In order for your processing actions to work well under the API Gateway configuration, and also to enable the use of contextual information that is relevant to the processing actions, a library of functions and templates is available that you can use to assist you in writing the XSLT or GatewayScript transformations. This library provides the following mechanisms:

- Access to input property values (input from the assembly editor).
- Access to context values (the API Gateway runtime context).
- Access to the runtime payload message and media-type.
- Ability to modify the payload message.
- Ability to end the policy execution with an error.

### About this task

---

You create an implementation for your user-defined policy in the IBM DataPower Gateway. The implementation must adhere to the following conventions:

1. You must create a main processing rule, and this rule will be the starting point for the policy. The rule name must start with the name of the policy (the value of the **name** property in the "info" section of the policy YAML file), followed by -main; for example *checkmembership-main*.
2. There are no restrictions on what actions the processing rule can execute, on condition that the rule adheres to the naming convention in point 1. and the instructions that are detailed in this topic.
3. The names of the processing actions and all other objects also must start with the name of the user-defined policy (the value of the **name** property in the "info" section of the policy YAML file).
4. If your processing rule runs transformation actions that use XSLT or GatewayScript files, these files must be stored in the following location: `local://policy/<policy-name>`.
5. If you have certificate and key files that must be stored in the `cert:` folder, the names of these files also must start with the name of the user-defined policy (as defined in the policy YAML file).
6. If your policy is using GatewayScript transformations, your policy code must require `apim.custom.js`, for example:

```
var apic = require('local://isp/policy/apim.custom.js');
```

The following steps describe how to create an implementation for your user-defined policy:

- [Create a main processing rule.](#)
- [Create one or more processing actions:](#)
  - [Configure access to input properties.](#)
  - [Configure access to the runtime context.](#)
  - [Configure access to the input payload.](#)
  - [Configure access to the HTTP headers.](#)
  - [Modify the payload in XSLT or gatewayscript.](#)
  - [Configure the implementation to produce error information.](#)
  - [Set variables.](#)
- [Export your policy implementation.](#)

## Procedure

---

1. Create a main processing rule that is called `<policy-name>-main` where `<policy-name>` is the name of your user-defined policy (the value of the **name** property in the "info" section of the policy YAML file).

Note: You must specify the following processing rule settings:

- Rule direction: Both Directions
- Non-XML Processing: on

2. Create one or more processing actions.

Each processing action should have a unique name and must start with the name of the user-defined policy (as defined in the policy YAML file).

Note: Different processing actions in DataPower might require different releases of the DataPower firmware. When using the GatewayScript actions, the minimum DataPower firmware that should be used is version 7.2.0.

The following processing actions are available:

- Configure access to the input properties in XSLT (`policyProperties`), or in GatewayScript (`apic.getPolicyProperty`).  
If required by the properties that are defined in the policy YAML file, when the policy is attached to an API the API developer must enter the input settings or variables for a particular property. This function provides the mechanism to retrieve these input settings or variables at run time. For an example code

snippet, see [Access to input properties code snippet](#).

- Configure access to the runtime context in XSLT (`getContext`), or in GatewayScript (`apic.getContext`).  
When an API is invoked, runtime information about the request can be accessed by using a user-defined policy, and this information is known as the runtime context. The function `getContext` provides the mechanism to access this runtime context. For an example code snippet, see [Access to runtime context code snippet](#). For a complete list of context variables, see [API Gateway context variables](#).
- Configure access to the input payload in XSLT (`payloadRead`), or in GatewayScript (`apic.readInput(callback)`).  
Some policies require access to the input message or payload that is provided by the application. Accessing the input message or payload during an assembly flow can be difficult, as this information might change as the policies and the assembly steps run. This function provides a mechanism to get the correct input message or payload at the time of policy execution. For an example code snippet, see [Access to input payload code snippet](#).

This function returns an XML node-set that contains the payload of the request. If the payload is in JSON format, a JSONx node-set is returned that can then be manipulated within an XSLT or GatewayScript stylesheet. If the payload is not in JSON or XML format, the node-set that is returned is empty.

Note: The `payloadType()` function can be called to determine what type of payload (XML or JSONx) will be returned by the `payloadRead()` or the `apic.readInput(callback)` function.

- Configure access to the HTTP headers in XSLT (`getContext`), or in GatewayScript (`apic.getContext`).  
The HTTP header information of a request can be retrieved from the request context. The HTTP header names are normalized to lowercase. The function `getContext()` provides the mechanism to access the HTTP header information. For an example code snippet, see [Access to HTTP headers code snippet](#).

Note: Access or modification of HTTP headers by using DataPower extensions, such as `dp:set-request-header`, is not advisable, as such actions might yield unexpected results when the policy is combined with other policies and assembly steps.

- Modify the payload in XSLT or GatewayScript.  
When a policy implementation is required to change a payload message, you must configure the DataPower Transform Processing Action to set the Output context field to `OUTPUT` (referred to as the `OUTPUT` named context).

The output must be an XML node-set, which represents an XML or SOAP message, or a JSON message by using JSONx. To assist the API Gateway policy framework to accept the new or transformed message, call the `apim-output` template. For an example code snippet, see [Modify the payload code snippet](#).

Note: The modify the payload processing action works only with a proxy task. An HTTP or web service task always overwrites the output of a policy implementation.

- Configure the implementation to produce error information.  
If you require your policy implementation to produce error information when there is a failure, you must configure the implementation by calling the error template. For an example code snippet, see [Configure error information code snippet](#).
- Set variables.  
Use the `setVariable` template to set a runtime variable to a specified string value. This value can then be retrieved by using the function `getVariable()`, or by mapping the value into a step. For an example code snippet, see [Set variables code snippet](#).

### 3. Export your policy implementation from DataPower.

From the DataPower user interface, export the objects and files that are referenced by your main processing rule as a compressed file. No other objects or files should be exported. The following file list is an example configuration of an export from DataPower:

```
Archive: checkmembership.zip
11819 11-19-2014 22:53 dp-aux/basetypes.xml
3028207 11-19-2014 22:53 dp-aux/drMgmt.xml
6456 11-19-2014 22:53 dp-aux/map-dmz.xml
7299 11-19-2014 22:53 dp-aux/management.xml
23130 11-19-2014 22:53 dp-aux/SchemaUtil.xml
216003 11-19-2014 22:53 dp-aux/clixform.xml
5284 11-19-2014 22:53 local/policy/checkmembership/check.xml
5061 11-19-2014 22:53 export.xml
```

The example shows the exported configuration file (`export.xml`) that contains all the processing actions and DataPower objects, the referenced files, and the DataPower configuration schemas that are created by using the configuration export in the DataPower user interface.

## Results

You have created an implementation for your user-defined policy and exported this implementation from the DataPower user interface.

## What to do next

Create a user-defined policy package that can then be imported into IBM API Connect. For details, see [Packaging and importing your policies into IBM API Connect](#).

## Related concepts

- [Authoring policies](#)

## Related information

- [IBM DataPower Gateway](#)

DataPower Gateway (v5 compatible) only

# Packaging and importing your policies into IBM API Connect

Make your user-defined policy available to API developers by packaging it and importing it into an IBM® API Connect Catalog.

## Before you begin

Before you can package a user-defined policy and import it into IBM API Connect, you must complete the following tasks:

1. [Describe your policy](#) in a YAML file.
2. [Implement your policy](#), by using DataPower processing rules and actions.

## Procedure

To package and import your user-defined policy, complete the following steps:

1. Create a .zip file that contains the following folder structure:

```
<policy-name>.yaml
implementation/
 policy-name.zip
```

where

- `<policy-name>.yaml` is your policy definition file. The YAML file must use the name that is specified in its `name` property.
- `implementation/policy-name.zip` is your policy implementation for a policy that is deployed to the DataPower Gateway. The policy implementation file contains the DataPower processing rules and actions that were exported from DataPower in step 2.

The name of the .zip file must start with the name of the user-defined policy (as defined in the policy YAML file). If the implementation requires certificate and key files, these files must be added to the implementation directory.

Note: Your package .zip file must contain a .zip policy implementation file.

2. Use the API Manager user interface to import your policy into a Catalog. For more information, see [Importing a user-defined policy into a Catalog](#).

Note:

- You must import your user-defined policy into every Catalog in API Manager that you require your policy to run in.
- For the policy to be displayed on the palette in the [API assembly editor](#), you must import the user-defined policy into the Sandbox Catalog.

## What to do next

Make your user-defined policy package accessible to the API Designer UI. To display user-defined policies in the API Designer UI, edit the `.apiconnect/config` file to point to the directory containing the policy.yaml file. You can edit the global config file (`~/apiconnect/config`) or a project-specific config file (`~/ProjectA/.apiconnect/config`). Follow these steps:

1. Create one or more directories to hold your policy.yaml files that are accessible to Node.js, and copy the user-defined policy package into this directory. For example, create directories named `/ProjectA/Policies` and `/ProjectB/Policies`.
2. Extract the policy package so that the folder structure is visible, for example `policy.yaml` and `implementation/policy-name.zip`. The file containing the custom policies must be named `policy.yaml`.
3. Add the `userPolicies: [ ]` array object to the config file in `~/ProjectA/.apiconnect/` or `~/apiconnect/`, and edit the array object so it contains the absolute path to the director(ies) containing the policy.yaml file.

Following is an example of an `~/apiconnect/config` file showing the `userPolicies` entry pointing to the directory containing the policy.yaml file:

```
apim_server: us.apiconnect.ibmcloud.com
us.apiconnect.ibmcloud.com.meta:
 formFactor: BLUEMIX_PUBLIC
 serverCapabilities:
 authTypes:
 - basic
 - token
 toolkit:
 version_recommended: 2.1.30
 version_minimum: 2.1.30
catalog: >-
 apic-catalog://us.apiconnect.ibmcloud.com/orgs/sampleorg-myspace2/catalogs/sb
app: >-
 apic-app://us.apiconnect.ibmcloud.com/orgs/sampleorg-myspace2/apps/acme-bank
userPolicies:
 - /ProjectA/Policies
```

Note:

- The `userPolicies: [ ]` array object contains a list of absolute paths to the policy director(ies). The values can point to a parent location, for example `userPolicies: [ "~/mypolicies" ]`. Alternatively, each policy location can be specified in the array, for example `userPolicies: [ "~/mypolicies/policya", "~/mypolicies/policyb" ]`. However, do not use both of these approaches in the same directory. For example, do not create policies in a directory structure like this `~/mypolicies`, `~/mypolicies/policya`, and `~/mypolicies/policyb`.
4. Open the API Designer UI. The user-defined policy is available on the policy assembly palette, and you can add the policy to your API definition.

DataPower Gateway (v5 compatible) only

## Reference

Reference information for authoring policies in IBM® API Connect.

- [Implementation code examples](#)  
Example code snippets to help the creation of a user-defined policy implementation.

DataPower Gateway (v5 compatible) only

## Implementation code examples

Example code snippets to help the creation of a user-defined policy implementation.

Note: If you are using GatewayScript, you must include the following command:

```
var apic = require('./apim.custom.js');
```

where *apic* is the common name used for the GatewayScript examples in this topic. However, *apic* could be any given name of your choice, for example you could use:

```
var apim = require('./apim.custom.js');
```

and then you would start your calls with *apim*.

- [Access to input properties code snippet](#)
- [Access to runtime context code snippet](#)
- [Access to input payload code snippet](#)
- [Access to HTTP headers code snippet](#)
- [Modify the payload code snippet](#)
- [Configure error information code snippet](#)
- [Set variables code snippet](#)

## Access to input properties code snippet

The following code block shows an example of how to access the input properties by using the XSLT `policyProperties()` function. The example defines a property that is named `a_property`, which is declared as an integer value, but is retrieved in XSLT as text.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:func="http://exslt.org/functions"
 xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

 <!-- Contains the APIM functions -->
 <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xml"
/>

 <xsl:template match="/">
 <xsl:variable name="p" select="apim:policyProperties()" />
 <xsl:message>
 The value of my input property is
 <xsl:value-of select="$p/a_property" />
 </xsl:message>
 </xsl:template>
</xsl:stylesheet>
```

If you are using GatewayScript, you must call the function:

```
apic.getPolicyProperty(propertyName)
```

where *propertyName* is the name of the input property that you want to access. If the input property name is blank, the action will return all input properties.

## Access to runtime context code snippet

The following code block shows an example of how to access the runtime context by using the XSLT `getContext()` function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:func="http://exslt.org/functions"
 xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

 <!-- Contains the APIM functions -->
 <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xml"
/>

 <xsl:template match="/">
 <xsl:variable name="client-id" select="apim:getContext('client.app.id')"/>
 <xsl:message>
 The calling application is
 <xsl:value-of select="$client-id" />
 </xsl:message>
 </xsl:template>
</xsl:stylesheet>
```

If you are using GatewayScript, you must call the function:

```
apic.getContext(varName)
```

where *varName* is the name of the context variable that you want to access.

For a complete list of context variables, see [API Gateway context variables](#).



## Access to input payload code snippet

The following code block shows an example of how to access the input payload by using the XSLT `payloadRead()` function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:func="http://exslt.org/functions"
 xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

 <!-- Contains the APIM functions -->
 <xsl:import
 href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl"
 />

 <xsl:template match="/">
 <xsl:variable name="input" select="apim:payloadRead()" />
 <xsl:message>
 The input payload is
 <xsl:copy-of select="$input" />
 </xsl:message>
 </xsl:template>

</xsl:stylesheet>
```

If you are using GatewayScript, you must call the function:

```
apic.readInput(callback)
```

A callback is required because the actual payload read is asynchronous. The callback method is called when the payload is ready.

This function returns an XML node-set that contains the payload of the request. If the payload is in JSON format, a JSONx node-set is returned that can then be manipulated within an XSLT or GatewayScript stylesheet. If the payload is not in JSON or XML format, the node-set that is returned is empty.

The following example shows how to use the `payloadType()` function to determine what type of payload (XML or JSONx) will be returned by the XSLT `payloadRead()` function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:func="http://exslt.org/functions"
 xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

 <!-- Contains the APIM functions -->
 <xsl:import
 href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl"
 />

 <xsl:template match="/">
 <xsl:variable name="payloadType" select="apim:payloadType()" />
 <xsl:message>
 <xsl:text>Payload type is [</xsl:text>
 <xsl:value-of select="$payloadType" />
 <xsl:text>]</xsl:text>
 </xsl:message>
 </xsl:template>

</xsl:stylesheet>
```

## Access to HTTP headers code snippet

The following code block shows an example of how to access the HTTP headers in XSLT by using the `getContext()` function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:func="http://exslt.org/functions"
 xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

 <!-- Contains the APIM functions -->
 <xsl:import
 href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl"
 />

 <xsl:template match="/">
 <xsl:variable name="content-type" select="apim:getContext('request.headers.content-type')" />
 <xsl:message>
 The request content type is
 <xsl:value-of select="$content-type" />
 </xsl:message>
 </xsl:template>

</xsl:stylesheet>
```

If you are using GatewayScript, you must call the function:

```
apic.getContext(request.headers.headerName)
```

where `headerName` maps to the name of the header you want to access.

Note: Access or modification of HTTP headers by using DataPower® extensions, such as `dp:set-request-header`, is not advisable, as such actions might yield unexpected results when the policy is combined with other policies and assembly steps.

## Modify the payload code snippet

The output from a user-defined policy must be an XML node-set, which represents an XML or SOAP message, or a JSON message by using JSONx. The following code block shows an example of how to modify the payload in XSLT. To assist the API Gateway policy framework to accept the new or transformed message, call the `apim-output` template, as shown in the following example.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:apim="http://www.ibm.com/apimanagement"
 xmlns:jsonx="http://www.ibm.com/xmlns/prod/2009/jsonx">

 <!-- Contains the APIM functions -->
 <xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl"
/>

 <xsl:template match="/">
 <!-- Creates a JSON document (empty object is for simplicity) -->
 <jsonx:object>
 </jsonx:object>

 <!-- Indicates the media type of the output being produced -->

 <xsl:call-template name="apim:output">
 <xsl:with-param name="mediaType" select="'application/json'" />
 </xsl:call-template>
</xsl:template>

</xsl:stylesheet>
```

where `mediaType`:

- `'application/json'` is when the output is written in JSONx format.
- `'application/xml'` is when the output is written in XML format.

If you are using GatewayScript, you must call the function:

```
apic.output(mediaType)
```

where `mediaType` is:

- `application/json` is when the output is written in JSONx format.
- `application/xml` is when the output is written in XML format.

Specifying the media type allows the next steps in the assembly flow to understand how to process the new payload.

Tip: The output from a user-defined policy must be XML or JSONx. JSONx is an IBM standard format to represent JSON as XML. One way to convert output GatewayScript JSON data into JSONx, is to add a `Convert Query Params to XML` action to follow the GatewayScript action within the same policy rule. The `Convert Query Params to XML` action must have an `Input Conversion` with the `Default Encoding` set to `JSON`. The output from the GatewayScript action must be the input for the `Convert Query Params to XML` action for JSONx to be produced.

## Configure error information code snippet

The following XSLT code block shows an example of how to configure the policy implementation to produce error information by calling the `apim-error` template.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:apim="http://www.ibm.com/apimanagement">

 <!-- Contains the APIM functions -->
 <xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl"
/>

 <!-- Indicates this policy has a failure and provides
 additional information for the client application -->
 <xsl:template match="/">
 <xsl:call-template name="apim:error">
 <xsl:with-param name="httpCode" select="'401'" />
 <xsl:with-param name="httpReasonPhrase" select="'Unauthorized'" />
 <xsl:with-param name="errorMessage" select="'Please select a Plan'" />
 </xsl:call-template>
 </xsl:template>

</xsl:stylesheet>
```

where:

- `httpCode` is the code of the required error message.
- `httpReasonPhrase` is the reason for the error.
- `errorMessage` is the suggested action for the user.

If you are using GatewayScript, you must call the template:

```
apic.error(name, httpCode, httpReasonPhrase, message)
```

where:

- **name** is the name of the error.
- **httpCode** is the code of the required error message.
- **httpReasonPhrase** is the reason for the error.
- **message** is the suggested action for the user.

## Set variables code snippet

The following XSLT code block shows an example of how to set a runtime variable to a specified string value by calling the `setVariable` template.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:func="http://exslt.org/functions"
 xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

 <!-- Contains the APIM functions -->
 <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl"
/>

 <xsl:template match="/">
 <xsl:call-template name="apim:setVariable">
 <xsl:with-param name="varName" select="'serviceEndpoint'" />
 <xsl:with-param name="value" select="'https://endpoint.host.com/data'" />
 </xsl:call-template>
 <xsl:message>
 <xsl:text>Variable [</xsl:text>
 <xsl:value-of select="'serviceEndpoint'" />
 <xsl:text>] set to [</xsl:text>
 <xsl:value-of select="'https://endpoint.host.com/data'" />
 <xsl:text>] </xsl:text>
 </xsl:message>
 </xsl:template>
</xsl:stylesheet>
```

where:

- **varName** is the name of the runtime variable that you want to set a value to.
- **value** is the string value that you want to set the variable to. This can be a literal value, or another variable. For example, to set your named variable to the value of the Content-Type header in a request, you would specify the **value** as `$(request.headers.content-type)`.

If you are using GatewayScript, you must call the template:

```
apic.setvariable(varName, varValue, action)
```

where:

- **varName** is the name of the runtime variable that you want to set a value to, or that you want to add or clear.
- **varValue** is the string value that you want to set the variable to. This can be a literal value, or another variable. For example, to set your named variable to the value of the Content-Type header in a request, you would specify the **varValue** as `request.headers.content-type`. This property is only required when **set** or **add** is specified as the action.
- **action** is the action that you want to apply to the variable. Valid options are:
  - **set**
  - **add**
  - **clear**

If no option is set, the default option of **set** is applied.

The following XSLT example shows how to retrieve the value of a runtime variable by using the `getVariable()` function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:dp="http://www.datapower.com/extensions"
 xmlns:func="http://exslt.org/functions"
 xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

 <!-- Contains the APIM functions -->
 <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl"
/>

 <xsl:template match="/">
 <xsl:variable name="varValue" select="apim:getVariable('serviceEndpoint')" />
 <xsl:message>
 <xsl:text>Variable [</xsl:text>
 <xsl:value-of select="'serviceEndpoint'" />
 <xsl:text>] = [</xsl:text>
 <xsl:value-of select="$varValue" />
 <xsl:text>] </xsl:text>
 </xsl:message>
 </xsl:template>
</xsl:stylesheet>
```

where

- **varValue** is the name of the runtime variable that you want to retrieve a value for.

If you are using GatewayScript, you must call the function:

```
apic.getvariable (varName)
```

where `varName` is the name of the runtime variable that you want to retrieve a value for.

## Related concepts

---

- [Authoring policies](#)

## Related tasks

---

- [Implementing your policy](#)



---

# Authoring policies for the DataPower API Gateway

A user-defined policy for the DataPower® API Gateway consists of a package containing configuration details that define the actions of the policy. You publish the package to the DataPower API Gateway to make it available to APIs that are deployed there.

There are two types of user-defined policies, *Catalog scoped* user-defined policies and *global scoped* user-defined policies:

Catalog-scoped user-defined policies

Catalog-scoped user-defined policies are available to APIs only in the Catalogs that you specify. Use a Catalog scoped user-defined policy if you want to limit the availability of your policy on a Catalog specific basis. The possible actions of a Catalog scoped user-defined policy are limited to the API Connect built-in assembly policies. For more information, see [Defining, packaging, and publishing a catalog-scoped policy for the API Gateway](#).

Global-scoped user-defined policies

Global-scoped user-defined policies are available to APIs in every Catalog in every provider organization. Use a global scoped user-defined policy in the following situations:

- You want to make your policy available everywhere rather than limiting its availability to specific Catalogs.
- Your policy uses a DataPower implementation, where configuration changes are made directly on the DataPower API Gateway.

For more information, see [Defining, packaging, and publishing a global-scoped policy for the API Gateway](#).

- [Defining, packaging, and publishing a catalog-scoped policy for the API Gateway](#)  
Create a catalog-scoped, user-defined policy that can be used in API assemblies within the catalog where the policy is published.
- [Defining, packaging, and publishing a global-scoped policy for the API Gateway](#)  
Create a global-scoped, user-defined policy that is available to APIs in any of the catalogs in any provider organization.

---

# Defining, packaging, and publishing a catalog-scoped policy for the API Gateway

Create a catalog-scoped, user-defined policy that can be used in API assemblies within the catalog where the policy is published.

## About this task

---

When a catalog-scoped policy is published, you can only use it in the API Assembler's Policy palette for APIs contained within the same catalog where the policy was published. Complete the following tasks to define, package, and publish the catalog-scoped policy.

Tip: During this task, you will create several zip files. On Mac OS X, run the `zip` command on the command line with the following flags:

```
-x ".DS_Store" -x "__MACOSX"
```

Including the flags prevents the auto-generated `.DS_Store` and `__MACOSX` files from being added to the zip.

- [Structure of the catalog-scoped user-defined policy YAML file](#)  
Review the required structure of the YAML file that contains your catalog-scoped user-defined policy.

---

## Step 1: Define a catalog-scoped policy for API Gateway

---

Use the DataPower Web GUI to define a new policy that can be used in API assemblies within the catalog where the policy is published.

### Procedure

1. Log in to the DataPower Web GUI.
2. Create a new "test" domain and then switch to it.  
This test domain will be used as a "sandbox" to build and export the custom policies.
3. Navigate to Assembly Function and create a new object:
  - a. Name the object `<org-name>_<catalogname>_udp-<policyname>_<version>`.  
Example: "testorg\_stagecatalog\_udp-testpolicy\_1.0.0". Use the same name for all of the objects that you create in this task.  
Note: This naming convention specifies the provider organization and catalog where the policy will be available, as well as the policy's name and version.
  - b. Under the "Assembly" property, create a new Assembly.
  - c. Under the new Assembly, create a new Rule.
  - d. Under the new Rule, create a new API Action.
  - e. Optionally add an implementation file for the new Action.

For example, an XSLT Action requires an XSL file. Create the file in the using the following path and name (where "udp" refers to user-defined policy):

```
local:/// <udp-name>/<udp-name>.<file-extension>
```

- f. Click Apply and then save all of the new objects.
4. Export the new user-defined policy.
  - a. Navigate to the newly created Assembly Function.
  - b. Click Export to export the new user-defined policy to a file named `<policyname>_<version>.zip`.
  - c. Open the exported `<policyname>_<version>.zip` file and confirm that all referenced files are included.
  - d. Delete the dp-aux folder from the zip.

This optional folder requires a lot of storage space and might cause a problem when you package and publish the new policy in the next task.

## Step 2: Package and publish a catalog-scoped policy for API Gateway

---

Decide how you want the new catalog-scoped policy to be managed, and then publish it to make it available to API assemblies within a particular catalog.

### About this task

Be sure to complete [Step 1: Define a catalog-scoped policy for API Gateway](#), before attempting to package and publish the new policy.

There are two methods for packaging and publishing a user-defined policy, based on how you want the policy to be managed:

- [Step 2, Option A: Package and publish a catalog-scoped policy directly in the DataPower configuration](#)
- [Step 2, Option B: Package and publish a catalog-scoped policy as a gateway extension that API Connect manages](#)

## Step 2, Option A: Package and publish a catalog-scoped policy directly in the DataPower configuration

---

Package and publish a catalog-scoped policy so it can be stored directly on the DataPower API Gateway.

### About this task

When you store a catalog-scoped policy on the DataPower API Gateway, the policy exists in the gateway's startup domain configuration and will be unaffected by the API Manager lifecycle. If you want to store the catalog-scoped policy directly on DataPower, you do not need to package it for API Manager. Instead, complete the following steps.

### Procedure

1. Import the custom policy created in [Step 1: Define a catalog-scoped policy for API Gateway](#) to the application domain for every DataPower node in the cluster.
2. Proceed to [Step 3: Create and publish a policy.yaml file](#), where you will create a file that describes the new policy so that API Manager can support it.

## Step 2, Option B: Package and publish a catalog-scoped policy as a gateway extension that API Connect manages

---

Package and publish a catalog-scoped policy as a gateway extension in API Connect.

### Before you begin

The xml-manager must be enabled on the default domain of every gateway that will have a gateway-extension published to it.

### About this task

When you publish a catalog-scoped policy as a gateway extension in API Connect, it is subject to the API Manager lifecycle. The individual assembly functions will not be added to the API Connect Gateway Service (only global-scoped policies are added to the service).

### Procedure

1. Rename the `<policyname>_<version>.zip` file (that you created in [Step 1: Define a catalog-scoped policy for API Gateway](#)) to `<org-name>_<cat-name>_<policyname>_<version>.zip`.  
The new file name indicates the provider organization and the catalog where the policy will be used.
2. Create a gateway-extension manifest.json file as explained in [Gateway extensions manifest](#).  
The manifest contains one or more entries. The following example contains a sample entry for the new catalog-scoped policy and references the new policy filename.

```
{
 "filename": "<org-name>_<cat-name>_<policyname>_<version>.zip",
 "deploy": "immediate",
 "type": "dp-import",
},
```

3. Create a new zip file called gateway-extension.zip that contains the following files:
  - manifest.json
  - `<org-name>_<cat-name>_<policyname>_<version>.zip`
4. Publish the new gateway extension using the API Conref Toolkit or the Cloud Manager UI:
  - API Connect Toolkit:  
Use the `apic gateway-extensions:create` command to publish the gateway extension as explained in the [apic gateway-extensions](#) topic.
  - Cloud Manager UI:  
Complete the following steps to publish the gateway extension from Cloud Manager:
    - Log in to Cloud Manager.

- Click Topology > ⋮ > Configure gateway extensions > Upload gateway-extension.zip file and upload your new gateway-extension.zip file.
5. Verify that the new catalog-scoped policy is available in the API Assembler's Policy palette.
    - a. Log in to API Manager and open the Catalog where you published the new catalog-scoped policy.
    - b. Create an API.
    - c. Open the API Assembler page and view the Policy palette; verify that your new catalog-scoped policy displays and can be added to the API.


## Step 3: Create and publish a policy.yaml file

Create a file that describes the properties of the catalog-scoped policy, and then publish it to make it accessible to API Manager.

### Procedure

1. Create a YAML file that describes the new catalog-scoped policy.
  - a. Create a file named `<policyname>.yaml`.
  - b. Paste the following content into the new file:

```
attach:
- rest
- soap
policy: 1.0.0
info:
 title: <policyname>
 name: <policyname>
 version: 1.0.0
 description: <description-here>
properties:
 $schema: 'http://json-schema.org/draft-04/schema#'
 type: object
 properties:
 exampleProperty:
 label: Test Property
 description: Enter any value
 type: string
 exampleProperty2:
 label: Another Test Property
 description: Enter any value
 type: string
 required:
 - exampleProperty2
gateways:
- datapower-api-gateway
```

- c. Modify the content as needed to correctly describe your new catalog-scoped policy.  
For more information, see [Structure of the catalog-scoped user-defined policy YAML file](#).
  - d. Save and close the file.
2. Create a new zip file called `<policyname>.zip` that contains the new YAML file.
  3. Publish the new policy using either the API Connect Toolkit or the API Manager:
    - API Connect Toolkit:  
Use the `apic policies:create` command to publish the gateway extension as explained in the [apic policies:create](#) topic.
    - API Manager:  
Complete the following steps to publish the new policy:
      - Log in to API Manager.
      - Click Manage Catalogs > Select Catalog > Gateways > selected\_gateway > ⋮ > View policies and upload your new policy.zip file.
  4. Verify that the new catalog-scoped policy is available in the API Assembler's Policy palette.
    - a. In API Manager, select the Catalog where you published the new catalog-scoped policy.
    - b. Create an API.
    - c. Click  (next to Save) and for the Target Catalog, select the catalog where you published the new policy.
    - d. Save the API.
    - e. Open the API Assembler page and view the Policy palette; verify that your new catalog-scoped policy displays and can be added to the API.

## Structure of the catalog-scoped user-defined policy YAML file

Review the required structure of the YAML file that contains your catalog-scoped user-defined policy.

The .yaml file for a catalog-scoped user-defined policy contains the following sections:

- [Specification version](#)
- [Information](#)
- [Attach](#)
- [Gateways](#)
- [Properties](#)
- [Assembly section](#)

### Specification version

Set the specification version by adding the following line to the beginning of the file:

policy: 1.0.0

## Information

---

Complete the information section with details about the policy by using the following syntax:

```
info:
 title: Title of policy
 name: shortnameofpolicy
 version: 1.0.0
 description: An example policy
 contact:
 name: name1
 url: url1
 email: email1
```

where:

- **title** is the title of the policy. Any string can be used, but the title should be kept short so that it can be displayed in the API Manager user interface.
- **name** is the short name for the policy. Must be a single word, and contain only alphanumeric characters, and the - (dash) and \_ (underscore) characters. The name is case-sensitive, and should be 20 characters or less so that it can be displayed in the API Manager user interface.  
Important: The policy short name must be different from the OpenAPI names of the built-in policies, otherwise it will cause a policy conflict in the API assembly definition. For a list of the OpenAPI built-in policy names, see [execute](#). Also, consider including an appropriate unique prefix or suffix string in the short names of your user-defined policy to prevent possible name conflicts with future built-in policies.  
Note: The **name** property must be identical to the Node.js module name.
- **version** is the version number of the policy. This is a reserved property.  
Tip: The *version.release.modification* version numbering scheme is recommended, for example *1.0.0*.  
Note: The **info.version** in policy.yaml must match the **version** indicated in package.json.
- **description** (optional) is a short description of the policy.
- **contact** (optional) is the contact information for the policy, where:
  - **name** (string) is the identifying name of the contact person or organization.
  - **url** (string) is the URL that points to the contact information.
  - **email** (string) is the email address of the contact person or organization.

For example:

```
info:
 title: Custom Invoke Policy
 name: custom-invoke
 version: 1.0.0
 description: Invokes httpbin
 contact:
 name: IBM DataPower Samples
 url: 'https://github.com/ibm-datapower/'
 email: steve-product-manager@ibm.com
```

## Attach

---

Complete the attach section by specifying which types of API flow the policy can be attached to.

- **rest** indicates that this policy can be attached to a REST API flow.
- **soap** indicates that this policy can be attached to a SOAP API flow.

The attach section must contain at least one API flow type. If a policy can be attached to both API flow types, they must both be indicated in the attach section.

For example:

```
attach:
- rest
- soap
```

## Gateways

---

Specify the DataPower® API Gateway, as follows:

```
gateways:
- datapower-api-gateway
```

## Properties

---

Complete the properties section by defining a JSON schema (based on JSON Schema Specification Draft 4, but rendered in YAML format) that contains the list of properties the policy will declare as required input. These properties are presented to the API author at development time, when the properties can be configured or mapped to values as required. The JSON schema defines a root object that contains the following JSON properties:

- **type**
- **properties**
- **required**

The syntax and definition of these properties matches the JSON schema definition. The following code block shows an example of a simple schema that defines one required property (*a\_property*):

```
properties:
 $schema: "http://json-schema.org/draft-04/schema#"
 type: object
 properties:
```

```

a_property:
 label: a label
 description: a description
 type: a type
 required:
 - a_property

```

The root `type` of the schema must be an `object`, as shown in the example schema. You can define zero or more properties in the schema. Required properties are declared by using the `required` parameter, as shown in the example schema. The policy will not be accessible in the API Manager assembly editor, unless all the required properties have values. Each property is an object with the following items:

- `label` is a short name for the property and is displayed in the Name column of the API Manager assembly editor.
- `description` is a short description for the property and is displayed in the Description column of the API Manager assembly editor.
- `type` must be one of the following primitives that are supported:
  - `integer`
  - `number`
  - `string`
  - `boolean` (this primitive is shown as a check box in the assembly editor)
  - `array`
- `default` (optional) specifies a default value for the property.
- `enum` (for properties with a `type` of `string`) is an array of valid values.

For example:

```

properties:
 $schema: 'http://json-schema.org/draft-04/schema#'
 type: object
 properties:
 test:
 label: Test Property
 description: Enter any value
 type: string
 required:
 - test

```

## Assembly section

The assembly section defines the assembly policy flow that will be activated when the user-defined policy is invoked. For example:

```

assembly:
 execute:
 - invoke:
 version: 2.0.0
 title: custom-invoke
 header-control:
 type: blacklist
 values: []
 parameter-control:
 type: whitelist
 values: []
 timeout: 60
 verb: keep
 cache-response: protocol
 cache-ttl: 900
 stop-on-error: []
 target-url: 'http://httpbin.org/headers'

```

You can determine the correct content of this section in either of the following ways:

- Define the assembly flow by using the graphical assembly editor in either the API Designer or API Designer user interface, then copy the `assembly` section from the Source tab. For more information on defining an assembly flow in the user interface, see [The Assemble view](#) and [API policies and logic constructs](#).
- Code the policies in the `execute` subsection of the `assembly` section by hand; for details, see [execute](#).

The assembly section can contain only the API Connect [built-in policies](#). However, you can include a [gatewayscript](#) policy that references native DataPower configuration.

## Defining, packaging, and publishing a global-scoped policy for the API Gateway

Create a global-scoped, user-defined policy that is available to APIs in any of the catalogs in any provider organization.

### About this task

When a global-scoped policy is published, you can use it in the API Assembler's Policy palette for APIs in any catalog belonging to any provider organization. Complete the following tasks to define, package, and publish the global-scoped policy.

Tip: During this task, you will create several zip files. On Mac OS X, run the `zip` command on the command line with the following flags:

```
-x ".DS_Store" -x "__MACOSX"
```

Including the flags prevents the auto-generated `.DS_Store` and `__MACOSX` files from being added to the zip.

### Step 1: Define a global-scoped policy for API Gateway

Use the DataPower Web GUI to define a new policy that can be used in API assemblies in all catalogs belonging to all provider organizations.



## Procedure

1. Log in to the DataPower Web GUI.
2. Create a new "test" domain and then switch to it.  
This test domain will be used as a "sandbox" to build and export the custom policies.
3. Navigate to Assembly Function and create a new object:
  - a. Name the object `<policyname>_<version>`.  
Example: "testpolicy\_1.0.0". Use the same name for all of the objects that you create in this task. Because the policy will be available in all provider organizations and catalogs, there is no need to specify that information in the object name.
  - b. Under the "Assembly" property, create a new Assembly.
  - c. Under the new Assembly, create a new Rule.
  - d. Under the new Rule, create a new API Action.
  - e. Optionally add an implementation file for the new Action.  
For example, an XSLT Action requires an XSL file. Create the file in the using the following path and name (where "udp" refers to user-defined policy):  
  
`local:///<udp-name>/<udp-name>.<file-extension>`
  - f. Click Apply and then save all of the new objects.
4. Export the new user-defined policy.
  - a. Navigate to the newly created Assembly Function.
  - b. Click Export to export the new user-defined policy to a file named `<policyname>_<version>.zip`.
  - c. Open the exported `<policyname>_<version>.zip` file and confirm that all referenced files are included.
  - d. Delete the dp-aux folder from the zip.  
This optional folder requires a lot of storage space and might cause a problem when you package and publish the new policy in the next task.

## Step 2: Package and publish a global-scoped policy for API Gateway

---

Decide how you want the new global-scoped policy to be managed, and then publish it to make it available to API assemblies.

### About this task

Be sure to complete the steps in [Step 1: Define a global-scoped policy for API Gateway](#) before attempting to package and publish the new global-scoped policy. There are two methods for packaging and publishing a user-defined policy, based on how you want the policy to be managed:

- [Step 2, Option A: Package and publish a global-scoped policy directly in the DataPower configuration](#)
- [Step 2, Option B: Package and publish a global-scoped policy as a gateway extension that API Connect manages](#)

## Step 2, Option A: Package and publish a global-scoped policy directly in the DataPower configuration

---

Package and publish a global-scoped policy so it can be stored directly on the DataPower API Gateway.

### About this task

When you store a global-scoped policy on the DataPower API Gateway, the policy exists in the gateway's startup domain configuration and will be unaffected by the API Manager lifecycle.

### Procedure

1. Import the `<policyname>_<version>.zip` file containing the new global-scoped policy to the application domain for every DataPower node in the cluster.  
Complete the following steps on every DataPower node in the cluster--if you omit a node, then the policy will not be available in the API Assembly's Policy palette.
  - a. On the DataPower node, navigate to the "API Connect Gateway Service" object.
  - b. Under the "User-defined policies" property, add a new Assembly Function by importing the `<policyname>_<version>.zip` file containing your new global-scoped policy.
  - c. Click Apply, and then Save.
2. Verify that the new global-scoped policy is available in the API Assembler's Policy palette.
  - a. Log in to API Manager.
  - b. Create an API.
  - c. Open the API Assembler page and view the Policy palette; verify that your new global-scoped policy displays and can be added to the API.

## Step 2, Option B: Package and publish a global-scoped policy as a gateway extension that API Connect manages

---

Package and publish a global-scoped policy as a gateway extension in API Connect.

### Before you begin

The xml-manager must be enabled on the default domain of every gateway that will have a gateway-extension published to it.

### About this task

When you publish a global-scoped policy as a gateway extension in API Connect, it is subject to the API Manager lifecycle.

### Procedure

1. Make a copy of the `<policyname>_<version>.zip` file containing your new global-scoped policy and remove the `<version>` number from the file name.  
This name change ensures that the relative path of the file meets the requirement to match the following regular expression: `implementation\[w_-\]+\ .zip`.  
You can retain the copy of the file with its original name for tracking purposes.

2. Create a local folder named implementation and store the renamed zip file in that folder.
3. Create a YAML file that describes the new global-scoped policy.
  - a. Create a file named `<policyname>.yaml`.
  - b. Paste the following content into the new file:

```
policy: 1.0.0
info:
 title: <policyname>
 name: <policyname>
 version: 1.0.0
 description: <description-here>
```

Only the `policy` and `info` sections are needed in the `<policyname>.yaml` for this task.

- c. Save and close the file.
4. Create a new zip file called `<policyname>_<version>.zip` that contains the following files:
  - The implementation folder
  - `<policyname>.yaml`
5. Create a gateway-extension manifest.json file as explained in [Gateway extensions manifest](#). The manifest contains one or more entries. The following example contains a sample entry for the new global-scoped policy and references the original policy `<policyname>_<version>.zip` file that you created in [Step 1: Define a global-scoped policy for API Gateway](#):


```
{
 "filename": "<policyname>_<version>.zip",
 "deploy": "immediate",
 "type": "user-defined-policy",
},
```

6. Create a new zip file called `gateway-extension.zip` that contains the following files:
  - `manifest.json`
  - `<policyname>_<version>.zip`

This is the file that you created in [step 4](#) of this task (it contains the implementation folder and the `<policyname>.yaml` file).
7. Publish the new gateway extension using one of the following methods:
  - API Connect Toolkit:
 

Use the `apic gateway-extensions: create` command to publish the gateway extension as explained in the [apic gateway-extensions](#) topic.
  - Cloud Manager UI:
 

Complete the following steps to publish the gateway extension from Cloud Manager:

    - Log in to Cloud Manager.
    - Click Topology. . Configure gateway extensions and upload your new `gateway-extension.zip` file.
8. Verify that the new global-scoped policy is available in the API Assembler's Policy palette.
  - a. Log in to API Manager.
  - b. Create an API.
  - c. Open the API Assembler page and view the Policy palette; verify that your new global-scoped policy displays and can be added to the API.

---

## Developer Portal: socialize your APIs

The Developer Portal is your one-stop-shop for sharing your APIs with application developers.

In addition to allowing application developers to find and subscribe to APIs, the Developer Portal provides additional features including forums, blogs, comments, and ratings, together with an administrative interface for customizing the Developer Portal to your brand and theme. See the following topics for more information.

### [Using the Developer Portal](#)

As well as enabling application developers to find and use your APIs, the Developer Portal also provides features such as API analytics, forums, blogs, and rating facilities.

### [Configuring the Developer Portal site](#)

Customize the look and feel of the Developer Portal site, and brand it for your organization.

### [Concepts in the Developer Portal](#)

Understand the various concepts and terminology that are referenced throughout the Developer Portal.

### [Getting started configuring the Developer Portal site](#)

A quick start guide to configuring your Developer Portal site.

### [Developer Portal tutorials](#)

Guided examples that take you through some of the most common user scenarios, as well as some of the more complex areas.

- [Using the Developer Portal](#)  
As well as enabling application developers to find and use your APIs, the Developer Portal also provides features such as API analytics, forums, blogs, and rating facilities.
- [Configuring the Developer Portal site](#)  
Customize the look and feel of your Developer Portal site, and brand it for your organization.
- [Managing the Developer Portal by using the toolkit CLI](#)  
You can perform operations on your Developer Portal by using the API Connect toolkit CLI.
- [Troubleshooting guide for the Developer Portal](#)  
Use this guide to help you diagnose and resolve Developer Portal issues in IBM® API Connect. For example, why is the Developer Portal not connecting to the Management server, why is my site not created, and why can't I log in?
- [Migrating your Developer Portal from Version 5 to Version 10](#)  
Guidance on how to migrate a Developer Portal site from IBM API Connect Version 5, to IBM API Connect Version 10.
- [Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10](#)  
Information about the upgrade of the Developer Portal from Drupal 9 to Drupal 10.

- [Developer Portal tutorials](#)  
Tutorials for using the Developer Portal.

---

## Using the Developer Portal

As well as enabling application developers to find and use your APIs, the Developer Portal also provides features such as API analytics, forums, blogs, and rating facilities.

The Developer Portal is a convenient place to share APIs with application developers. After a Developer Portal has been enabled through the API Manager, and one or more API Products have been published, application developers can browse and use the APIs. Typically, a Developer Portal is customized to fit the corporate branding and design requirements of a particular organization. For more information about configuring a Developer Portal site, see [Configuring the Developer Portal site](#). However, when the Developer Portal is first enabled it can be used as is, for example for test and development purposes, or for internal use only.

The following topics provide information about the main API consumer concepts in the Developer Portal, and how an application developer can navigate the default site. However, bear in mind that the actual consumer experience might be very different depending on how the site is configured. For tutorials guiding you through enabling a Developer Portal site, and navigating and using APIs as an application developer, see [Tutorial: Creating the Developer Portal](#) and [Tutorial: Creating Accounts and Applications on the Developer Portal](#).

- [Logging in as a consumer to the Developer Portal by using the CLI](#)  
Use the toolkit CLI to log in to the Developer Portal as a consumer.
- [Exploring APIs and Products in the Developer Portal](#)  
As an application developer, you can browse, test, and subscribe to APIs in the Developer Portal by exploring the available API Products.
- [Applications in the Developer Portal](#)  
You can view, edit, configure, and create applications in the Developer Portal.
- [Analytics in the Developer Portal](#)  
You can view analytics for APIs in the Developer Portal at the application and organization levels. This information is displayed in dashboard views that show the analytics metrics in the form of *visualizations* (represented as charts).
- [Consumer organizations in the Developer Portal](#)  
Consumer organizations provide a way of grouping application developers together. You can add, remove, and configure Consumer organizations in the Developer Portal.
- [User accounts, passwords, and support in the Developer Portal](#)  
You can change general elements such as user accounts and passwords in the Developer Portal.

---

## Related information

- [IBM API Connect overview](#)

---

## Logging in as a consumer to the Developer Portal by using the CLI

Use the toolkit CLI to log in to the Developer Portal as a consumer.

---

## Before you begin

To log in as a consumer, you must have created a developer account in the Developer Portal as explained in the following tutorial: [Tutorial: Creating Accounts and Applications on the Developer Portal](#).

---

## About this task

You can use the toolkit CLI to log in to the Developer Portal as a consumer to view it and test how it behaves for your customers.

---

## Procedure

1. Set up the toolkit as explained in [Installing the toolkit](#).
2. Determine the name of your identity provider.  
When you log in from the CLI, you must provide the name of the realm (identity provider) that hosts your API Connect user account. If you don't know the name of your account's realm, you can look it up by using the following command.

In the CLI, run the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

Note that `<catalog_identity_provider_name>` is typically made up of `catalog_name-idp`, for example `sandbox-idp`.

3. In the toolkit CLI, run the following command to log in to the Developer Portal:  
Command syntax:

```
apic login --mode consumer --server consumer_endpoint_api --username username --password password --realm realm
```

Where:

- `consumer_endpoint_api` is the endpoint of the consumer API.
- `username` is your API Connect username.
- `password` is the password that corresponds to your API Connect username.
- `realm` is the user registry realm for the specified `<username>`.

Example:

```
apic login --mode consumer --server consumer.9a6e-88fa49bc.eu-gb.apiconnect.cloud.ibm.com -u my_username -p Blue123 -r consumer:frankenstein:om-portal/portal-idp
```

```
Logged into consumer.9a6e-88fb51bc.eu-gb.apiconnect.cloud.ibm.com successfully
```

---

## Exploring APIs and Products in the Developer Portal

As an application developer, you can browse, test, and subscribe to APIs in the Developer Portal by exploring the available API Products.

### Browsing available APIs

In API Connect API providers package their APIs into Products to offer one or more APIs to application developers. The providers then use Plans to control access to APIs and to manage API usage. Products are packages that contain both the APIs and the accompanying Plans.

From the Developer Portal home screen, click API Products to browse the available products. You can then select a Product to explore the available Plans and APIs further, including viewing API code snippets, and request and response examples.

Note: If you have an account on the Developer Portal, you can specify a default programming language for the code snippets to display in by editing the Code Snippet Language in your account settings.

### Subscribing to a Plan

A Plan is a collection of API operations or subsets of operations from one or more APIs. In order to subscribe to a Plan, you must have an account on the Developer Portal and have created an App. For more information, see [User accounts, passwords, and support in the Developer Portal](#) and [Applications in the Developer Portal](#). Rate limits specify how many requests an App is allowed to make during a specified time interval. A Plan can have a shared rate limit for all the operations, or each operation can have a different rate limit. In addition, Plans can have multiple rate limits set per Plan and per operation, at second, minute, hour, day, and week time intervals. Plans can also have burst limits to prevent usage spikes that might damage infrastructure. Multiple burst limits can be set per Plan, at second and minute time intervals.

After you have identified the Plan that you want to use, click Subscribe and then select the App that you want to use with this Plan. If the Plan is not restricted, you can use it immediately. If the Plan is restricted, the subscription is shown as Pending Approval, and you cannot use the requested Plan until the administrator approves your request.

You cannot subscribe to a monetized Plan without first adding your payment information to your Consumer organization. You can register your payment information by editing the Billing tab for your Consumer organization. Note: you must either be the Consumer organization owner, or have `Settings: Manage permissions` for your organization to be able to edit the billing information. Consumer organizations can have at most one payment method. You can delete your payment method and add a different method, for example if your credit card expires. If you change your subscription to a Plan that has a different pricing structure, the subscription billing system will add a prorated amount to the next invoice of your subscription billing cycle.

### Testing an API

You can test an API in the Developer Portal by using the interactive API document test tool. If the API operation that you want to test doesn't require a client ID, then you can use the test tool without the need to log in. For more information, see [Testing an API by using the Developer Portal test tool](#).

### Calling an API in your application

When you have subscribed to a Plan and you begin coding your application, you must retrieve the operation URL in order to call the API. You can retrieve the operation URL from the API overview page. For more information, see [Calling an API](#).

- [How to use a URL to navigate directly to a Product or API in the Developer Portal](#)  
You can navigate to a specific Product or API in the Developer Portal by using `x-ibm-name:version` in the URL.
- [Testing an API by using the Developer Portal test tool](#)  
You can test the behavior of an API without the need to write any code by using the Developer Portal test tool. You provide the necessary API parameters within the test tool and click Invoke to see the response.
- [Calling an API](#)  
When you have a selected a Plan and you begin coding your application, you must retrieve the operation URL to call the API.

---

## How to use a URL to navigate directly to a Product or API in the Developer Portal

You can navigate to a specific Product or API in the Developer Portal by using `x-ibm-name:version` in the URL.

To find a Product or API in the Developer Portal you typically navigate there by searching in the UI, clicking on the relevant Product, and then selecting the API. However, if you know the `x-ibm-name:version` of your Product or API, you can insert that information into the URL to go there directly.

To navigate directly to a specific API, use the following URL syntax:

```
portal.site.url/productselect/api_x-ibm-name:version
```

For example:

```
https://my.portal.com/myorg/sandbox/productselect/petstore-header-clientid:1.0.0
```

To navigate directly to a specific Product, use the following URL syntax:

```
portal.site.url/product/product_x-ibm-name:version
```

For example:

```
https://my.portal.com/myorg/sandbox/product/my-product:1.0.0
```

To navigate directly to a specific API within a specific Product, use the following URL syntax:

```
portal.site.url/product/product_x-ibm-name:version/api/api_x-ibm-name:version
```

For example:

```
https://my.portal.com/myorg/sandbox/product/my-product:1.0.0/api/petstore-header-clientid:1.0.0
```

---

# Testing an API by using the Developer Portal test tool

You can test the behavior of an API without the need to write any code by using the Developer Portal test tool. You provide the necessary API parameters within the test tool and click Invoke to see the response.

---

## Before you begin

To use the Developer Portal test tool with an API that requires an App client ID, you must first complete the following tasks:

- Create an App. For more information, see [Registering an application](#).
- Subscribe to a Plan that contains the API that you want to test. For more information, see [Exploring APIs and Products in the Developer Portal](#).

---

## About this task

The Developer Portal test tool is an interactive API document test tool. If the operation that you want to test does not require a client ID, then you can use Developer Portal test tool without the need to sign in. However, if the operation that you want to interact with requires a client ID, then you must sign in to the Developer Portal first.

Using the Developer Portal test tool is subject to the rate limit that is applied to an operation or Plan. For example, if an operation has a rate limit of 10 requests per minute and you invoke the operation, the number of requests that can be made is reduced to nine. The limitations are triggered every time that you click invoke within the minute interval. This caveat affects the quota of the App that is selected to use with the Developer Portal test tool, but it does not affect the quota of the other Apps that are using the same operation or Plan.

To test an API in the Developer Portal, the Allow this API to be tested check box must be selected in the API Manager. For more information, see [Creating API definitions](#).

Restriction:

- There are security mechanisms that prevent you using the Developer Portal test tool to test an Implicit or Authorization Code grant type in an OAuth provider API. Other grant types in the same OAuth provider API **can** be tested. If your OAuth provider allows those actions to function correctly from the test tool, you can disable this option. For more information, see [Disabling test tool restrictions](#).
- You cannot use the Developer Portal test tool with suspended applications.
- You can only test an unenforced API if `testable=true`, and if the existing API implements CORS and is using HTTPS.

Note: GraphQL APIs can be tested from the Developer Portal user interface. However, not all authentication methods are supported. You must use one of the following authentication methods:

- Unsecured
- Client ID, passed in the header
- Client ID and secret, passed in the header

Also, `cost` metrics are not supported, and rate limits specific to GraphQL are not listed in Product plans.

---

## Procedure

1. To use the Developer Portal test tool with an API that does not require the client ID of an App, complete the following steps:

- a. Click API Products.  
All of the APIs that can be used by application developers are displayed.
- b. Click the name of the API that you want to test.
- c. Select an operation and then select Try It.
- d. Click Try this operation.
- e. Supply the values for required headers or parameters.
- f. If the operation is secured with Basic Authentication, supply the credentials.
- g. Click Send Request.

The result is displayed in the Response Body field. You can continue to test different parameter values as necessary.

Note:

The first time that you click Try It, you might be presented with a security error. Copy the URL from the Request URL field, open it in a browser window, and accept the security certificate. You are not presented with the security error again.

Also, for performance reasons, if the payload is large, more than 1500 DOM highlighted elements, you get the response, but it is not pretty printed.

2. To use the Developer Portal test tool with an API that does require the client ID of an App, complete the following steps:

- a. Sign in to the Developer Portal.
- b. If you have not already done so, you must create an App so that you can test an API that requires a client ID. For more information, see [Registering an application](#).
- c. Ensure that your App is subscribed to the Plan that contains the API that you want to test. For more information, see [Exploring APIs and Products in the Developer Portal](#).
- d. Click API Products.  
All of the APIs that can be used by application developers are displayed.
- e. Click the API that you want to test.
- f. Select the **Provide credentials for Client ID (API Key)** operation and then select Try It.  
The API Key Identification window displays.
- g. Use the Register an application in order to select a client ID drop-down list to select an App to test the API with.
- h. Click Save credentials.
- i. If a client secret is required, enter the value in the Client Secret field.
- j. Find the operation that you want to test, then click Try It.
- k. Supply the required parameters and values.
- l. If the operation is secured with Basic Authentication, supply the credentials.
- m. Click Send.

The result is displayed in the Response section. You can continue to test different field values as necessary.

Note: The first time that you click Invoke, you might be presented with a security error. Click the link that is provided to accept the security certificate. You are not presented with the security error again.

## Related tasks

- [Calling an API by using CORS](#)

## Calling an API

When you have a selected a Plan and you begin coding your application, you must retrieve the operation URL to call the API.

## About this task

Be aware of the following points when you call APIs in IBM® API Connect:

- In the HTTP response message for status code 200 the reason phrase is replaced with OK.
- API error messages are displayed in English only.

## Procedure

To retrieve the operation URL, complete the following steps:

1. Click API Products, then select a product.
2. Click the API that you want to work with.  
The API overview page opens.
3. Select the operation you need and then copy the endpoint.

This is the URL that the application calls, and its structure is defined:

Type of API	URL
REST APIs	<code>https://host/org/catalog/api/operation</code>
SOAP APIs	<code>https://host/org/catalog/api</code> for all operations in the WSDL.

where:

- *host* is the fully qualified host name of your gateway cluster.
  - *org* is the URL path of your organization.
  - *catalog* is the name of your Catalog.
  - *api* is the name of your API.
  - *operation* is the URL path of your operation.
4. Take note of any parameters, the request body, and the response body. Code your application to create the expected requests and handle the expected responses.

Depending on the Identify your application using setting for the API, you might have to provide a client ID, or client ID and client secret. To do this, complete the following steps:

5. To find the client ID, complete the following steps:
  - a. Click Apps, then click the application name that you want to work with.
  - b. Select the Show check box for Client ID.  
The client ID is displayed.
  - c. Provide the client ID with the header parameter `&client_id=`  
For example, the URL used in the API might be:

```
https://host/org/catalog/api/quote?loanAmount=20000
```

but when you call it with a client ID of 1234, change the URL to:

```
https://host/org/catalog/api/quote?loanAmount=20000&client_id=1234
```

Note: API Gateway only A client ID is generated automatically by API Connect when you register an application. However, if you specify a customized client ID, by using the CLI or REST API, then the length must not exceed 512 bytes, otherwise the gateway will reject the API request and return a 401 error.

6. The client secret is produced when you register an application. Provide the client secret with the query parameter `&client_secret=`. If you did not note the client secret when you registered the application, you must reset it; for information, see [Managing applications](#).

The client ID, or client ID and secret can be logged along with the URL. Web servers usually log the URL in their access logs, which would give away the client secret. If you do not want to expose your client ID or secret in the URL, complete the following steps.

7. For the client ID, set the header, **X-IBM-Client-Id**, as part of the HTTP message that the application sends when it calls the API.  
An example URL statement might be:

```
curl --header "X-IBM-Client-Id: 1234" https://host/org/catalog/api/quote?loanAmount=20000
```

Note: API Gateway only A client ID is generated automatically by API Connect when you register an application. However, if you specify a customized client ID, by using the CLI or REST API, then the length must not exceed 512 bytes, otherwise the gateway will reject the API request and return a 401 error.

8. For the client secret, set the header, **X-IBM-Client-Secret**, as part of the HTTP message that the application sends when it calls the API.  
For example, the URL would be:

```
https://host/org/catalog/api/quote?loanAmount=20000
```

and set the following HTTP headers:

```
X-IBM-Client-Id=1234
X-IBM-Client-Secret=ABCD
```

## What to do next

---

Monitor the API and application usage. For more information, see [Managing applications](#).

- [Calling an API by using CORS](#)

CORS (cross origin resource sharing) is a technique that allows calls to be made from code that is running in a browser to a third-party server (such as APIs running on an API Connect Gateway). These calls are, by default, not allowed as per the same origin security policy that is applied to the browser sandbox. Without CORS support, web developers are required to use more complex techniques such as server-side proxies.

---

## Calling an API by using CORS

CORS (cross origin resource sharing) is a technique that allows calls to be made from code that is running in a browser to a third-party server (such as APIs running on an API Connect Gateway). These calls are, by default, not allowed as per the same origin security policy that is applied to the browser sandbox. Without CORS support, web developers are required to use more complex techniques such as server-side proxies.

---

## About this task

API Connect Gateway servers support CORS to make it as easy as possible for web developers to use APIs within their web applications.

CORS is supported in the following browsers:

- Chrome 3+
- Firefox 3.5+
- Internet Explorer V11, or later
- Opera 12+
- Safari 4+

A CORS enabled browser automatically sends either a simple CORS request, consisting of the original request with the addition of the **Origin** header, or a preflight request followed by a simple CORS request.

An example CORS preflight request is as follows:

```
OPTIONS /org/env/api/resourceHTTP/1.1
User-Agent: useragent details
Access-Control-Request-Method: GET
Access-Control-Request-Headers: header names
Host: x.xx.xxx.xx
Origin: https://example.com
Accept: */*
```

You do not need to create CORS requests yourself, other than for testing or troubleshooting purposes.

A CORS response is received from the gateway; for example:

```
HTTP/1.1 200 OK
X-Backside-Transport: FAIL FAIL
Connection: Keep-Alive
Transfer-Encoding: chunked
Access-Control-Allow-Origin: https://example.com
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: accept, accept-language, content-type, x-ibm-client-id
Access-Control-Allow-Methods: methods allowed on the resource
Vary: Origin
```

---

## Applications in the Developer Portal

You can view, edit, configure, and create applications in the Developer Portal.

Before you can subscribe to a Plan that contains the API or APIs that you want to use, you must first register an application by creating an App. You can create more than one App, and these Apps can be edited and deleted as required.

To learn about applications in the Developer Portal, use the following topic links.

- [Registering an application](#)  
Before you can use an API, you must register your application to the Developer Portal.
- [Managing applications](#)  
You can show or reset a client ID for an application, verify or reset a client secret, and view the details of the APIs in the application. You can also unsubscribe from a Plan that the application is subscribed to.
- [Editing an application](#)  
You can edit applications in the Developer Portal. If a user has the correct permissions, they can edit the content of any application in their organization.
- [Deleting an application](#)  
You can delete applications in the Developer Portal.
- [Changing an application image](#)  
You can change the image for an application in the Developer Portal.
- [Verifying an application client secret](#)  
You can verify an application client secret in the Developer Portal

---

## Registering an application

Before you can use an API, you must register your application to the Developer Portal.

## About this task

---

When you register an application, you are provided with a client ID and client secret for the application. You must supply the client ID when you call an API that requires you to identify your application by using a client ID, or a client ID and client secret.

You can, optionally, add further client ID/client secret pairs to an application, any of which can be used to identify the application when calling an API.

## Procedure

---

To register an application, complete the following steps:

1. Click Apps.  
The Apps page opens.
2. Click Create new app.
3. Optional: Specify a URL in the Application OAuth Redirect URL(s) field.  
You can specify multiple OAuth redirect URLs by clicking Add another item. If only one redirect URL is specified, and the application does not provide the `redirect_uri` in the OAuth request, then API Connect automatically uses the one redirect URL specified. However, if more than one redirect URL is specified, then the application must provide the `redirect_uri` in the OAuth request, or the OAuth request is rejected.
4. Click Submit.  
Your application is displayed.
5. The client secret is hidden; click the Show check boxes, make a note of your client secret because it is displayed only once.  
You must supply the client secret when you call an API that requires you to identify your application by using a client ID and client secret.  
Note: The client secret cannot be retrieved. If you forget it, you must reset it.
6. Optional: In the Certificate field, paste the public X509 certificate for your application, in PEM format.  
The Certificate field is available only if the APIs that have been published to the Developer Portal include at least one API that has been secured with TLS mutual authentication. You must complete this field if you want to call an API that has been secured with TLS mutual authentication. For more information, see [Composing a REST API definition](#).  
  
These certificates are sent to the management server and to the gateway, and are used for certificate verification when the application calls an API that has been secured with TLS mutual authentication.  
  
Important: When you paste the certificate, ensure that it contains no line breaks, and that the (-----BEGIN CERTIFICATE-----) prefix and (-----END CERTIFICATE-----) suffix are removed.
7. Optional: The client ID is hidden, to display the client ID for your application, click Subscriptions, then select the Show check box for Client ID.  
The client ID is displayed and can be hidden again by clearing the check box.
8. Optional: To verify your client secret, click Subscriptions, then click Verify, enter your client secret in the Secret field, then click Submit.  
You have confirmed whether your client secret is correct or incorrect.
9. To add an additional client ID and client secret to the application, complete the following steps:
  - a. Click Subscriptions.
  - b. Click Add alongside Credentials.  
The "Add new application credentials" window opens.
  - c. Enter a name and an optional summary, and click Submit.
  - d. Select the Show Client ID or Show Client Secret check box to display the client ID or client secret for the new credentials.
  - e. To add a description to a set of client credentials, or to change the current description, click Update alongside the required credentials.
  - f. To remove a set of client credentials from the application, click Delete alongside the required credentials.If you add additional credentials to an application, any of the associated client ID/client secret pairs can be used to identify the application when calling an API.  
Note: If you add two or more sets of client credentials to an application, OAuth tokens are not shared between them; each client credential set uses a different OAuth token.
10. To add an image, click Upload image from the application overflow menu.  
A new window opens; click Browse, select an image from your directory, and click Submit.
11. Optional: To change the URL that authenticated OAuth flows for this application should be redirected to, click Edit from the application overflow menu, then enter a URL in the Application OAuth Redirect URL(s) field.
12. Optional: To change the application name or description, click the Edit from the application overflow menu.

## Results

---

Your application is registered.

## What to do next

---

Select a Plan to use with your application, see [Exploring APIs and Products in the Developer Portal](#).

## Related information

---

- [IBM API Connect overview](#)
- [Creating an API key security definition](#)

## Managing applications

---

You can show or reset a client ID for an application, verify or reset a client secret, and view the details of the APIs in the application. You can also unsubscribe from a Plan that the application is subscribed to.



## About this task

---

On the application details page, you can display and reset a client ID and client secret. You might want to reset a client secret if it is compromised or you forget it. A client secret is only displayed once, so after you reveal it for the first time at a time of your choosing, you can use this window to verify that what you have is correct or reset it if necessary.

You can, optionally, add further client ID/client secret pairs to an application, any of which can be used to identify the application when calling an API.

## Procedure

---

To manage an application, complete the following steps:

1. Click Apps.  
The Apps page opens.
2. Select the application that you want to work with from the displayed list.
3. Click Subscriptions.
4. The client ID is hidden; to display the client ID, select the Show check box. The client ID is displayed, and can be hidden again by clearing the Show check box.
5. To reset your client ID, click Reset.  
Warning: API calls made by the application with the existing client ID will fail.
6. If the client secret is compromised or you forget it, click Reset to generate a new value for the client secret.
7. If you are unsure whether the value that you have for the client secret is correct, you can click Verify, enter the value, and click Submit to confirm whether or not it is correct.
8. To add an additional client ID and client secret to the application, complete the following steps:
  - a. Click Subscriptions.
  - b. Click Add alongside Credentials.  
The "Add new application credentials" window opens.
  - c. Enter a name and an optional summary, and click Submit.
  - d. Select the Show Client ID or Show Client Secret check box to display the client ID or client secret for the new credentials.
  - e. To add a description to a set of client credentials, or to change the current description, click Update alongside the required credentials.
  - f. To remove a set of client credentials from the application, click Delete alongside the required credentials.If you add additional credentials to an application, any of the associated client ID/client secret pairs can be used to identify the application when calling an API.  
Note: If you add two or more sets of client credentials to an application, OAuth tokens are not shared between them; each client credential set uses a different OAuth token.
9. To view more details about an individual API, click the API name.  
The operations are listed.
10. To unsubscribe from a Plan, click Unsubscribe alongside the Plan name.

## Related information

---

- [IBM API Connect overview](#)

## Editing an application

---

You can edit applications in the Developer Portal. If a user has the correct permissions, they can edit the content of any application in their organization.

## Procedure

---

1. From the Developer Portal home page, click Apps.
2. Select the target application.
3. Click Edit from the application overflow menu.
4. Make any required changes, then click Submit.

## Deleting an application

---

You can delete applications in the Developer Portal.

## Procedure

---

1. From the Developer Portal home page, click Apps.
2. Select the required application.
3. Click Delete from the application overflow menu.
4. Click Delete again to confirm.

## Changing an application image

---

You can change the image for an application in the Developer Portal.

## Procedure

---

1. From the Developer Portal home page, click Apps.
2. Select the required application.
3. Click Upload image from the application overflow menu.
4. Browse for the target image, then click Submit.  
You can remove the image by clicking Remove.

---

## Verifying an application client secret

You can verify an application client secret in the Developer Portal

### Procedure

---

1. From the Developer Portal home page, click Apps.
2. Select the target application.
3. Click Subscriptions.
4. Click Verify for the client secret.
5. Enter your secret string and click Submit.  
You receive a notification saying whether or not the secret that you entered is correct.

---

## Analytics in the Developer Portal

You can view analytics for APIs in the Developer Portal at the application and organization levels. This information is displayed in dashboard views that show the analytics metrics in the form of *visualizations* (represented as charts).

Restriction:

- If the Catalog that is associated with this Developer Portal has two or more gateway services enabled then, for analytics data to be available, the gateway services must all be associated with the **same** analytics service. If the set of associated analytics services for the gateway services includes two or more different analytics services then the Developer Portal does **not** display analytics data. An analytics service is associated with a gateway service in the Cloud Manager user interface; see [Associating an analytics service with a gateway service](#).
- If the visualizations in your dashboard views display no data, show data only for a time period in the past, or the Dashboard tab does not show up, a possible reason might be that access to analytics data within API Connect has been switched off. Contact your API provider for confirmation.

Use the following topic links to learn how to access the various analytics available to you in the Developer Portal:

- [Application analytics in the Developer Portal](#)  
From the Developer Portal, you can view interactive analytic information about all of the APIs used by an application.
- [Organization analytics in the Developer Portal](#)  
From the Developer Portal, you can view interactive analytic information about all of the APIs within an organization.

---

## Application analytics in the Developer Portal

From the Developer Portal, you can view interactive analytic information about all of the APIs used by an application.

### About this task

---

All members of the consumer organization can view analytics information relating to APIs used by an application.

### Procedure

---

1. Click Apps from the Developer Portal dashboard.
2. Click the name of the application for which you want to view analytic information.  
From the Dashboard tab, you can see your application analytics including, API Stats, Total Calls, and Total Errors.

---

## Organization analytics in the Developer Portal

From the Developer Portal, you can view interactive analytic information about all of the APIs within an organization.

### About this task

---

All members of the consumer organization can view analytics information relating to the APIs that are used within an organization.

### Procedure

---

1. Click your userName from the Developer Portal dashboard.
2. From the drop-down menu, select My organization.
3. Click the Analytics tab.

From the Analytics tab, you can see your organization analytics including, API Stats, Total Calls, and Total Errors.

---

## Consumer organizations in the Developer Portal

Consumer organizations provide a way of grouping application developers together. You can add, remove, and configure Consumer organizations in the Developer Portal.

Developer Portal users must be a member of at least one Consumer organization. A Consumer organization can have multiple members and multiple applications. Members must register an application before they can subscribe to an API. A Consumer organization owns all the applications that are created within it. The members do not own any applications. As long as there is one member of a Consumer organization, all applications continue unchanged.

Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: **A user already exists with this email address.**

To learn more about Consumer organizations in the Developer Portal, see the following topics.

- [Adding a Consumer organization from within the Developer Portal](#)  
You can create a new Consumer organization from within your user account in the Developer Portal. The new Consumer organization appears also in the API Manager UI after it is created.
- [Editing the name of a Consumer organization](#)  
You can edit the name of your Consumer organization from within the Developer Portal.
- [Switching Consumer organizations](#)  
You can switch from one Consumer organization to another in the Developer Portal.
- [Adding users to a Consumer organization](#)  
If the API Provider organization administrator has granted you permission to collaborate, you can invite users to join your Consumer organization. Those organization members can then access the Developer Portal and use the APIs that have been made available to the Consumer organization.
- [Removing a user from a Consumer organization](#)  
You can remove a developer from a Consumer organization in the Developer Portal.
- [Changing the ownership of a Consumer organization](#)  
You can change the ownership of a Consumer organization in the Developer Portal.
- [Deleting a Consumer organization](#)  
You can delete a Consumer organization in the Developer Portal.

---

## Adding a Consumer organization from within the Developer Portal

You can create a new Consumer organization from within your user account in the Developer Portal. The new Consumer organization appears also in the API Manager UI after it is created.

### Procedure

1. Sign in to your user account in the Developer Portal.
2. Click the arrow next to your Consumer organization name on the Developer Portal home page.
3. Select Create organization from the drop-down menu.
4. In the Organization Title field, type the name of the new Consumer organization, then click Submit.

### Results

Your new Consumer organization is created, and you can switch to it by clicking on its name in the Organization drop-down menu on the home page.

---

## Editing the name of a Consumer organization

You can edit the name of your Consumer organization from within the Developer Portal.

### Procedure

1. Sign in to your user account in the Developer Portal.
2. Click the arrow next to your Consumer organization name on the Developer Portal home page, and click My organization.
3. In the Manage tab, click the vertical ellipsis Manage organization icon, and click Edit organization.
4. Enter the new name for your Consumer organization in the Organization Title field, and then click Submit.  
You changed the name of your Consumer organization.

---

## Switching Consumer organizations

You can switch from one Consumer organization to another in the Developer Portal.

## Procedure

---

1. If you are a member of more than one Consumer organization, click the menu of Consumer organizations in the Developer Portal home page.
2. Select the Consumer organization that you want to switch to, and ensure that any applications you register are registered to the correct catalog.

## Adding users to a Consumer organization

---

If the API Provider organization administrator has granted you permission to collaborate, you can invite users to join your Consumer organization. Those organization members can then access the Developer Portal and use the APIs that have been made available to the Consumer organization.

### About this task

---

Depending on the permissions enabled by the Provider organization administrator, members can view applications and application activity, create and edit applications, manage client keys and subscribe to API Plans, or perform all activities.

Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: **A user already exists with this email address.**

## Procedure

---

To invite users to join your Consumer organization, complete the following steps.

Note: The ability to invite users is only visible if the API Provider organization administrator has granted you permission to collaborate.

1. Sign in to your user account in the Developer Portal.
2. Click the arrow next to your Consumer organization name on the Developer Portal home page, and click My organization.  
The organization page is displayed.
3. On the Manage tab, select Invite.
4. Enter the email address of the new user in the Email field.
5. Under the Assign Roles heading, use the radio buttons to select a role for the new user, the roles are:
  - **Administrator**-- Can administer the Consumer organization, including invite new members, and can create, and edit applications, manage client keys, and subscribe to API Plans.
  - **Developer**-- Can create and edit applications, manage client keys, and subscribe to API Plans.
  - **Viewer**-- Can only view applications and application activity.For full details of the permissions that are assigned to the Developer Portal roles, see [API Connect user roles](#).
6. Click Submit.  
An invitation is sent to the new user.

## Results

---

The user is added to the Consumer organization list of members, and they are sent an email inviting them to join the organization. The user must click the link that is provided in the email to activate their account, and complete the setup.

## Removing a user from a Consumer organization

---

You can remove a developer from a Consumer organization in the Developer Portal.

### Before you begin

---

You must be the owner of your Consumer organization and have previously added the developer you wish to remove from the organization. Note that removing a user from a Consumer organization does not delete their user account, it only removes them from that organization.

## Procedure

---

1. Click your user name in the Developer Portal home page.
2. Click My organization in the menu.
3. Click Remove user for the user you want to remove.

## Changing the ownership of a Consumer organization

---

You can change the ownership of a Consumer organization in the Developer Portal.

### Before you begin

---

You can change the ownership of a Consumer organization only if you are the owner of that organization. The user that you transfer ownership to must already be a member of the organization. To have an account on a Developer Portal, you must be a member of at least one Consumer organization.

## About this task

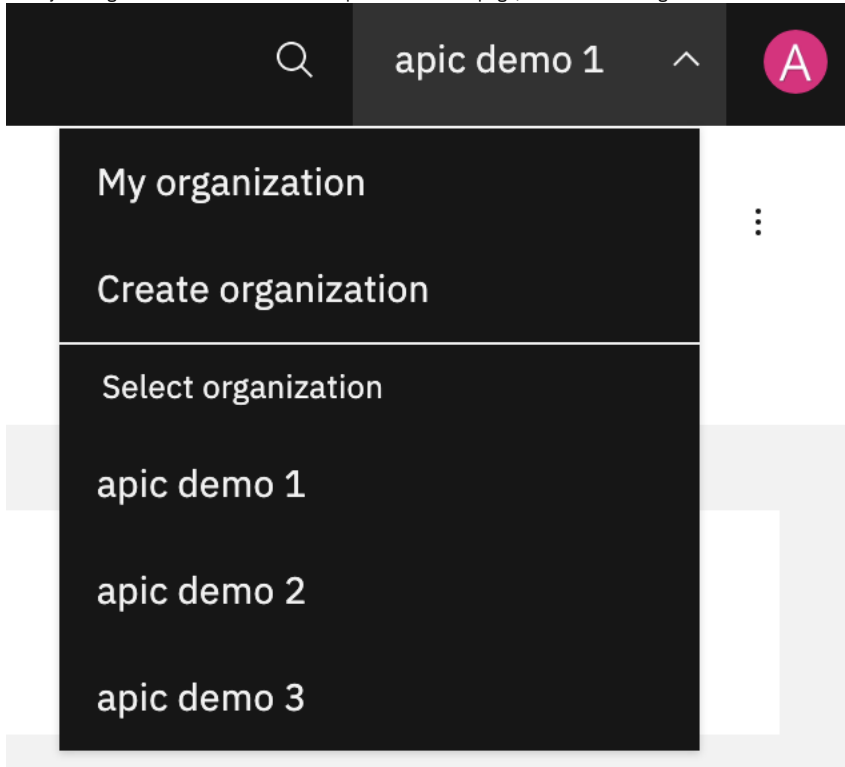
---


Transferring the ownership of a Consumer organization to another organization member, changes your role in that organization to App Developer.

## Procedure

---

1. Click your Organization name in the Developer Portal home page, and select the organization whose owner you want to change by using the drop-down arrow.



2. Click your Organization name in the Developer Portal home page, and select My organization.
3. Click the options icon, , and select Change ownership.

## Update the consumer organization owner

New Owner

apic demo user

Assign Previous Owner's Role

Administrator

Developer

Viewer

Cancel

Save

4. Select the **New Owner** by using the drop-down arrow.
5. Use the radio buttons to **Assign Previous Owner's Role**, then click Submit.

### Results

You successfully transferred ownership of a Consumer organization in the Developer Portal.

## Deleting a Consumer organization

You can delete a Consumer organization in the Developer Portal.

### Before you begin

You can delete a Consumer organization only if you are the owner of that organization. To have an account on a Developer Portal, you must be a member of at least one Consumer organization. Therefore, if you are a member of only one Consumer organization, to delete that organization you must first either join or create another Consumer organization, or you can delete your whole developer account. For more information, see [Deleting your developer account](#).

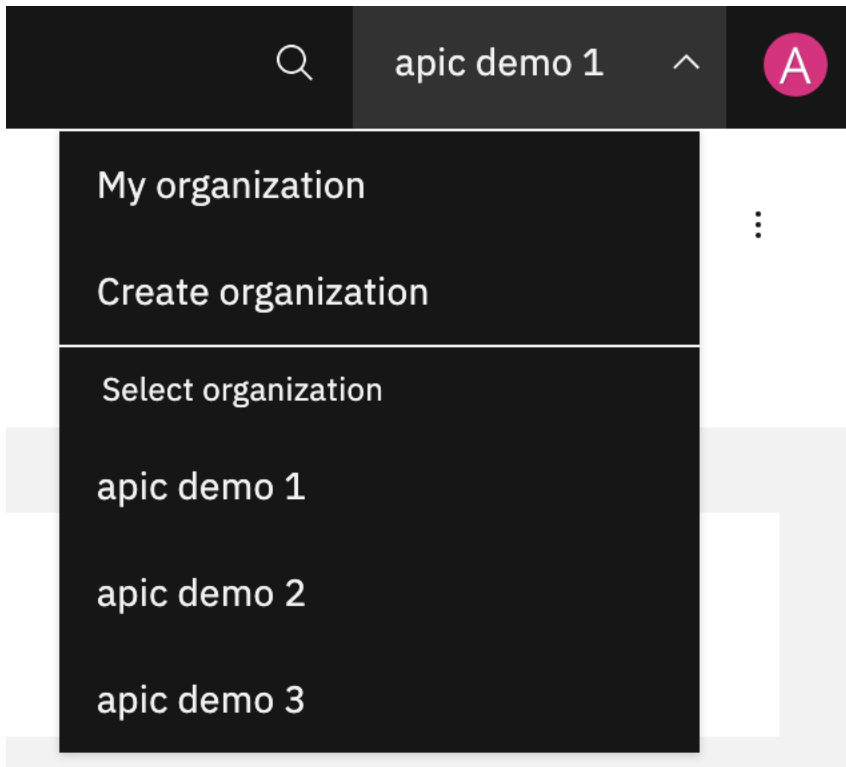
### About this task


Deleting a Consumer organization in the Developer Portal permanently removes access to the organization, and all of its applications and subscriptions, for all members of the organization. Consider carefully the impact on any members of your Consumer organizations before you delete the organization. Any users that were members of only the deleted organization, must create a new Consumer organization when they next logon to the Developer Portal.

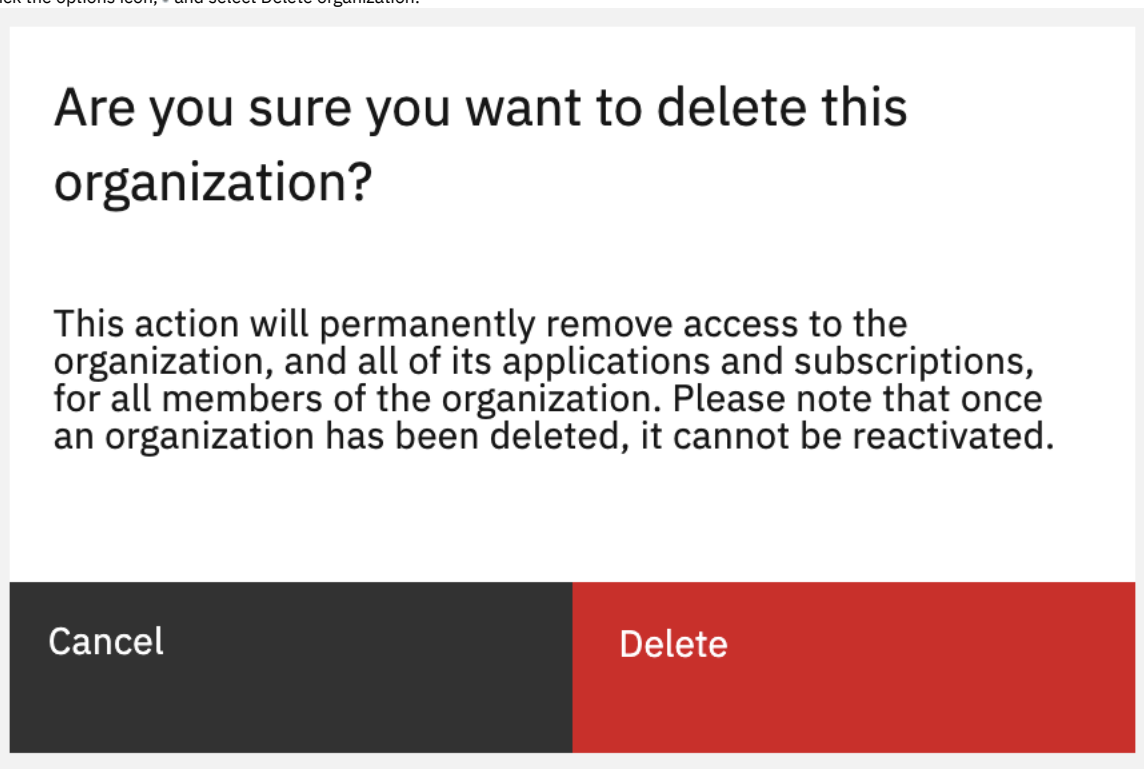
Important: When an organization is deleted, it cannot be reactivated. You might want to consider changing the ownership of a Consumer organization, rather than deleting it. For more information, see [Changing the ownership of a Consumer organization](#).

### Procedure

1. Click your Organization name in the Developer Portal home page, and select the organization that you want to delete by using the drop-down arrow.  
You must be in more than one Consumer organization.



2. Click your Organization name in the Developer Portal home page, and select My organization.
3. Click the options icon, , and select Delete organization.



4. Click Delete.

## Results

---

You permanently deleted a Consumer organization in the Developer Portal.

---

## User accounts, passwords, and support in the Developer Portal

You can change general elements such as user accounts and passwords in the Developer Portal.

Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: **A user already exists with this email address.**

To learn about general user features of the Developer Portal, see the following topics:

- [Creating a new developer account](#)  
You can create new developer accounts, and their associated Consumer organizations, in the Developer Portal.
- [Changing your account settings](#)  
You can edit your user account settings in the Developer Portal, such as changing your name, password, time zone, and language settings.
- [Deleting your developer account](#)  
You can delete your account in the Developer Portal.

---

## Creating a new developer account

You can create new developer accounts, and their associated Consumer organizations, in the Developer Portal.

### Before you begin

You need to provide an email address to create a new account. That email address must be a valid email address, and the domain that is used is required to have a valid MX record that is determined by DNS. You can disable the security module that checks that the email domain has a DNS record.

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click **Manage** to display it.
3. Navigate to **Extend > Disable** module.
4. In the search bar enter **check\_dns**, then select the **check\_dns** module and click **Disable**.
5. Click **Disable** to confirm that you want to disable the module.

### About this task

In order to use the Developer Portal, you must have a developer account, and be a member of at least one Consumer organization. When you create a developer account, a Consumer organization is created at the same time. You can be an owner, and a member of, multiple Consumer organizations. For more information about Consumer organizations, see [Consumer organizations in the Developer Portal](#).

Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: **A user already exists with this email address.**

### Procedure

1. From the Developer Portal home page, click **Create account**.
2. Populate the available fields with the details of the developer account and Consumer organization that you want to create.
3. Click **Sign up**.  
The email address that you provided for the new developer account is sent an email that contains an activation link for the new developer account. Use this link to activate your new account.

---

## Changing your account settings

You can edit your user account settings in the Developer Portal, such as changing your name, password, time zone, and language settings.

### About this task

You can edit various details about your user account, for example, change your password, update your code snippet language, change your time zone, select your preferred site language, and select an avatar.

### Procedure

1. Click your user name in the Developer Portal home page, and click **My account**.
2. To update your password, click **Change Password**, and update the fields as required.
3. To update other elements of your account, such as your name, time zone, and language, click **Edit** and update the fields as required.
4. Click **Save** to save your changes.

---

## Deleting your developer account



You can delete your account in the Developer Portal.

## Before you begin

---

You must be the owner of the developer account that you want to delete, and you must not own more than one Consumer organization. If you do own more than one Consumer organization, you must either change the ownership of those organizations, or delete them, before you can delete your account. If you own only one Consumer organization, when you delete your account the Consumer organization is deleted at the same time as your account.

## About this task

---

When you delete your account, any Consumer organizations that you own must also be deleted or transferred to new owners. If there are other members in a Consumer organization that you delete, these members will no longer have access to that organization, or any of its applications and subscriptions. Consider carefully the impact on any members of your Consumer organizations before you delete your account. For more information, see [Deleting a Consumer organization](#). Important: When an organization has been deleted, it cannot be reactivated. You might want to consider changing the ownership of your Consumer organizations before you delete your account. For more information, see [Changing the ownership of a Consumer organization](#).

## Procedure

---

1. Click your user name in the Developer Portal home page, and select My account.
  2. Click the Edit tab.
  3. Click Delete account.
  4. Click Delete to confirm deletion of the account.
- This action deletes your developer account and, if you own a Consumer organization, this organization is also deleted.

## Results

---

You successfully deleted your developer account in the Developer Portal.

## Configuring the Developer Portal site

---

Customize the look and feel of your Developer Portal site, and brand it for your organization.

In IBM® API Connect Version 10 the Developer Portal is based on the open source Drupal 10 content management system, and consequently it is almost infinitely customizable. What an individual user is able to configure, depends on the role the user has been assigned, and the associated permissions of that role. A user can be assigned one of four Developer Portal roles:

### Authenticated user

An authenticated user is, typically, an application developer, and can perform all of the actions that are described in the [Using the Developer Portal](#) section.

### Forum Moderator, Content Author, or Administrator

In addition to performing all of the tasks that an Authenticated user role can, a Forum Moderator, Content Author, or Administrator role have additional permissions related to configuring the Developer Portal site. For more information, see [Working with roles in the Developer Portal](#).

Note: By default, the Developer Portal administration operations are available only if you have Administrator access.

Before you start to configure your Developer Portal site, you should familiarize yourself with the various concepts and terminology that is referenced throughout the Developer Portal. See [Concepts in the Developer Portal](#).

When you are ready to start configuring your Developer Portal site, see [Getting started configuring the Developer Portal site](#) for a quick start guide on customization. There is also a tutorial that you can follow that takes you through an example customization scenario; see [Tutorial: Creating a custom theme for the Developer Portal](#).

If you are migrating from Version 5 to Version 10, see [Migrating your Developer Portal from Version 5 to Version 10](#) for guidance about how to transfer any specific customizations, such as custom modules and themes, from the Drupal 7 baseline to the Drupal 10 baseline.

In addition to the previous options, Developer Portal configuration operations are described in detail in the following sections:

### [Appearance](#)

Control and configure the general appearance of your Developer Portal site by setting and configuring a default theme.

Important: Editing API Connect themes, modules, or Drupal core on the filesystem is not permitted or supported. For more information, see [Creating a sub-theme](#) and [Extend](#).

### [Content](#)

Control multiple content entities in the Developer Portal, including the customization and restriction of specific content.

### [Structure](#)

Control the layout of the Developer Portal, such as configuring the front page, adding and changing blocks, and displaying Products and APIs in categories.

### [Configuration](#)

As an administrator, you can control multiple aspects of the Developer Portal configuration, including managing security and general configuration tasks.

### [People](#)

You can manage and customize the user experience in the Developer Portal, by altering permissions and assigning roles to specific users.

### [Forums](#)

Create and control forums in the Developer Portal.

### [Reports](#)

You can view reports about the general configuration and status of your Developer Portal site.

### [Extend](#)

You can extend the Developer Portal by creating and installing custom modules, or installing additional Drupal contributed modules.

## Related information

---

- [🌐 Drupal website](#)

---

## Concepts in the Developer Portal

The Developer Portal is built on Drupal 9, an open source content management technology. A good understanding of Drupal concepts and terminology enhances your ability to work with the Developer Portal.

In Drupal, most content on the website is treated as variations on the concept of a **node**. Node types, such as static pages, blog posts, and news items, are all stored in the same way. The navigation structure of the site is designed separately by editing **menus**, **views**, and **blocks**. Finally, the **theme** of the system controls the overall appearance of the site to visitors. Access to these depends on the permissions that your role possesses. These are explained more fully in the following sections.

---

### Nodes

A node is a set of individual related content such as a page, a poll, an article, a forum topic, or a blog post. For example, when you create a blog post, and define the body text, you also define its title, content, author link, creation date, and taxonomy. Some of these elements are shown by the theme layer when the node is displayed, and some elements are metadata that control where and when the node is displayed at all. Each set of content in a Developer Portal site is a node. You can apply new features or changes to all content of a single node type.

---

### Fields

A field is somewhere that you can add extra metadata to a node. The following types of data are examples of fields:

- Title
- Body
- Comment body
- Tags
- Image

You can create different types of fields, and they can be defined at the content-type level for content items and comments, at the vocabulary level for taxonomy terms, and at the site level for user accounts. Field types are defined by modules, and managed by the Field module. In addition, other modules might enable fields to be defined for their data.

---

### Content types

A content type is a predefined collection of fields. Content types define the default fields that content editors use to add content in a Developer Portal site, and help structure the authoring and developing of content. Content types can be displayed in the Developer Portal. You can control which content types, and the order and format that they are displayed, in the Developer Portal.

---

### Themes

You can use themes to control the overall appearance of your Developer Portal site. You can modify the appearance of the theme in following ways:

1. Extend the code of an existing theme.
2. Identify and use a theme that is provided by the Drupal community or a third-party website, and modify the theme settings.
3. Create a complete custom theme from scratch.

However, you cannot directly edit the API Connect theme as editing is not supported. Any edited versions of these files are overwritten when a fix pack or iFix is installed. So, the way to create a custom theme is to create a custom subtheme of the standard API Connect theme that the Developer Portal uses by default. A subtheme inherits all of the settings of its parent theme, apart from the settings that are overridden. For more information, see [Creating a sub-theme](#).

Most themes use the Twig template engine for PHP.

---

### Regions

The specific areas of a Developer Portal site in which content can be placed. Regions are customized and styled in the theme.

---

### Blocks

Blocks are boxes of content that can be displayed in regions on your Developer Portal page. Blocks can be made available to your Developer Portal site by enabling specific modules. After a block is created, its appearance can be modified. You can also define which Developer Portal page or pages blocks appear on. Some modules can provide multiple blocks when they are enabled, while other modules might not define new blocks. For more information, see [Adding and changing the blocks displayed on Developer Portal pages](#).

---

### Modules

Modules are similar to the concept of plug-ins, in that they extend the core functions of the Developer Portal site. A set of modules is implemented by default with the Drupal core, and there are extra modules that can be enabled to extend the default functions. You can find and add more modules to your Developer Portal site from the internet. For more information, see [Extend](#).

---

### Views

You can use views to manipulate the content that is displayed on a page, block, and other visual elements in the Developer Portal site. Views can be used along with content types to format the appearance of your site to your specification. For more information, see [Using the Views module in the Developer Portal](#).

## Pages

---

Pages are used by the features that can modify the appearance of the Developer Portal to customize the appearance of your Developer Portal site. Pages can be customized to be as specific as you require, and can be configured to satisfy the context of the situation that they are used in.

## Users

---

After you log in to the Developer Portal site, you have a user record in the Developer Portal database. User ID 1 is always reserved for the administrator account. Regardless of whether other user accounts are using remote authentication such as LDAP, the administrator account is always a local account to enable administration of the Developer Portal site. For more information, see [People](#).

## Roles

---

A role is a collection of permissions that define the actions that a user can do in the Developer Portal site. Users can be given one or more roles. By default, a user that is logged in to the Developer Portal is in the Authenticated role. Other roles that can be assigned to a user include:

- Administrator
- Forum Moderator
- Content Author

You can also create new custom roles. For more information, see [Working with roles in the Developer Portal](#).

## Permissions

---

Permissions define the actions that a user can or cannot do in a Developer Portal site. Permissions are additive. If a permission that enables a user to do an action is not assigned, the user cannot do the action. If a user has multiple roles, and any of them contain a specific permission, the user is able to do that action. There are also permissions that have security implications, and you are recommended to assign those permissions to trusted roles. For more information, see [Controlling access to Developer Portal content](#).

## Templates

---

Template files define how the output of a particular component can look. They are formatted as Twig template files. The following are examples of where you can use templates:

- `html.html.twig` - the template for Developer Portal HTML pages
- `page.html.twig` - the body of Developer Portal HTML page
- `node.html.twig` - the template for all of the content nodes
- `comment.html.twig` - the template for all of the comment nodes
- `search-result.html.twig` - the template for search results
- `node--product.html.twig` - the template for nodes that use the Product content type
- `node--product--teaser.html.twig` - the template for the previews for the Product content type

For more information about overriding templates in the Developer Portal, see [Applying a modified content type template](#).

## Related information

---

- [Understanding Drupal](#)
- [Theming Drupal](#)

## Getting started configuring the Developer Portal site

---

A quick start guide to the main tasks involved in configuring the Developer Portal to your brand or theme.

The Developer Portal is based on the open source Drupal 9 content management software, and consequently is almost completely customizable. The following sections take you through the most common tasks to get started with branding the Developer Portal to your organizational requirements, including links to the related topics in the main table of contents. Many more customizing options are described in the main table of contents, and the best way of finding help with a specific customization in this content, is to use the documentation search bar.

Important:

All of the customizations that are detailed in this topic are preserved when you update to new versions of API Connect. Drupal updates, as well as new functionality, defect fixes, and security updates, are made available through API Connect update packages as determined by IBM. These update packages can be applied by upgrading your deployment; see the upgrade instructions for your platform in [Installing and maintaining your IBM API Connect cloud](#). To find out what update packages are available, select your product on [IBM Fix Central](#).

Directly editing any API Connect themes, modules, or Drupal core on the file system is not permitted or supported, as edited versions of these files are overwritten when an update package is installed. Customizations must be done by using the proper mechanisms as described in this topic, such as custom modules and themes, and not by directly editing the source.

For more information about the API Connect support policy, see [IBM® API Connect Support Lifecycle Policy](#).

- [Create a custom theme](#)
- [Changing the page layout](#)
- [Changing the front page](#)
- [Advanced theme development](#)
- [Custom modules](#)
- [Hiding the admin login form](#)

- [What to do next - exploring the main configuration content](#)

---

## Create a custom theme

---

If you want your Developer Portal to have your corporate branding and style, you need to create a custom theme. Themes are composed primarily of Cascading Style Sheets (CSS) files, although they can include much more; see [Advanced theme development](#).

Directly editing the API Connect theme is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed. The way to create a custom theme is to create a custom sub-theme of the standard API Connect theme that the Developer Portal uses by default. A sub-theme inherits the parent theme's resources, and this means that your custom sub-theme CSS file needs to contain only the changes or overrides that you want to make from the default theme. The CSS file can contain as little or as many updates as you like.

For more information, see [Creating a sub-theme](#). You can also follow a tutorial that takes you through how to use the theme generator, customize a theme, and install a custom theme; see [Tutorial: Creating a custom theme for the Developer Portal](#).

While experience with Drupal is beneficial, the only skill you really need to create a sub-theme is CSS.

---

## Changing the page layout

---

While the theme defines *regions* of the page, such as a footer, or sidebar first, the page layout is defined by configuring which blocks appear in which regions. Blocks are controlled by using the Structure > Block layout options on the Developer Portal administrator dashboard. You can then further limit those blocks to show only on specific pages, or show on all pages except certain ones, to show only for certain roles, or for certain languages, and more.

For information about how to configure the page layout, see [Adding and changing the blocks displayed on Developer Portal pages](#).

Changing the page layout is done entirely in the Developer Portal UI as an Administrator.

---

## Changing the front page

---

The front page is a special case and does not use the blocks system that is mentioned previously. Instead, you configure it from the Structure > Pages options on the Developer Portal administrator dashboard.

For information about how to configure the front page, see [Configuring the front page](#).

Further options that you might want to consider when designing your front page include:

- Change the banner block content or image: [Changing the front page banner block](#).
- Display featured API Products by creating a featured content block: [Changing the front page Featured Content block](#).
- Create custom blocks with your own HTML content: [Adding and changing the blocks displayed on Developer Portal pages](#).
- Set up the social block to include your organization Twitter feed: [Integrating Twitter data into the social block](#).
- Add a carousel to the front page to display a series of images: [Implementing an image carousel](#).

Again, changing the front page is done entirely in the Developer Portal UI as an Administrator.

---

## Advanced theme development

---

The DOM structure for specific blocks and parts of the page is controlled by templates. These templates are Twig files that define the actual HTML output for a particular piece of content. You can override a Twig template by copying the original template into the templates folder of your custom sub-theme, and then editing the template to your specification. The Developer Portal then uses your template in preference to the original one. For more information, see [Applying a modified content type template](#).

Modifying Twig templates allows very fine-grained control over the structure of pages. However, it also means you might miss new features and defect fixes that are made to the original templates, as your Developer Portal is using your overrides instead of the originals. So, if you override a template, it is your responsibility to check the templates in the latest API Connect releases, and to ensure that any equivalent changes that are needed to your overrides to maintain functional equivalence are made.

If you're developing custom templates, knowledge of Twig templating language, Drupal, HTML, and CSS is recommended.

---

## Custom modules

---

If you need to modify the functional behavior of the Developer Portal, then you can create a custom module to do so. Drupal has an extensible programmatic API that is built around a system of *hooks*. By using these hooks in your custom modules you can change what happens when, for example, forms are displayed, forms are submitted, or make extra variables available to custom templates, and other options.

For more information on writing custom modules, see [Extend](#). You can also follow a tutorial that includes using a custom module, see [Tutorial: Adding validation to a field on the sign-up form](#).

Custom modules are written in PHP, and so Drupal experience and PHP knowledge are very beneficial.

---

## Hiding the admin login form

---

If your Developer Portal is public facing, you might want to hide the admin registry as an option on the login form. You can do this simply from the administrator dashboard, by clicking Configuration > System > IBM API Connect, and selecting Hide the admin registry on the login form. For more information, see [Hiding the admin registry on the login form](#).

---

## What to do next - exploring the main configuration content

---

There are an almost infinite number of configuration options available to you on the Developer Portal. Explore the following sections to find out more:

- [Appearance](#) - control general appearance elements, and learn how to create a custom theme.
- [Content](#) - add, configure, and control content elements, such as meta tags, custom pages, custom content types, and adding content to an API or Product.

- [Structure](#) - control how the content in your Portal is displayed, for example configuring the front page, changing block display, and using the Views module.
- [Configuration](#) - configure content moderation, perform general configuration tasks, and manage Developer Portal security.
- [People](#) - create and customize site configuration roles, control access to sites and to content, and create internal Consumer organizations to allow customization of specific API properties.
- [Forums](#) - create and control forums.
- [Reports](#) - list of available reports about the general configuration and status of your site.
- [Extend](#) - introduction to and instructions for creating custom modules, including information about developing in PHP, and how to use hooks in custom modules.
- [Developer Portal tutorials](#) - guided examples taking you through the simple and the more complex areas of configuration.

---

## Appearance

Control and configure the general appearance of your Developer Portal site by setting and configuring a default theme.

To learn how to control the appearance of your Developer Portal site, use the following topic links.

- [Controlling general appearance elements in the Developer Portal](#)  
You can control general appearance aspects of the Developer Portal.
- [Adding custom JavaScript to a custom theme](#)  
To increase customization of the Developer Portal, you can add custom JavaScript to custom themes that you install.
- [Applying a modified content type template](#)  
You can apply a modified content type template to override a default content type template in the Developer Portal. For example, you can override the product content type template to change its layout.
- [Creating a sub-theme](#)  
Create a custom theme for your Developer Portal site by generating and configuring a sub-theme. A sub-theme is a theme that inherits all the resources of a specified parent theme. You can then override specific resources to configure your required customizations.
- [Installing additional themes](#)  
You can install additional themes in the Developer Portal.
- [Deleting themes](#)  
You can delete a theme from your Developer Portal site.

---

## Controlling general appearance elements in the Developer Portal

You can control general appearance aspects of the Developer Portal.

To learn how to control general aspects of the Developer Portal appearance, use the following topic links:

- [Changing the shortcut icon](#)  
You can change the shortcut icon, or favicon, in the Developer Portal.
- [Changing the site email address](#)  
You can change the site email address in the Developer Portal.
- [Changing the site logo](#)  
You can change the site logo for your Developer Portal.
- [Changing the site name or site slogan](#)  
You can change the site name, or the site slogan, for your Developer Portal.
- [Changing the visibility of site branding](#)  
You can toggle the visibility of your site branding elements for your Developer Portal.

---

## Changing the shortcut icon

You can change the shortcut icon, or favicon, in the Developer Portal.

---

## Before you begin

You must have administrator access to complete this task.

---

## About this task

The shortcut icon, or favicon, is displayed in the address bar or bookmarks of most browsers. Drupal provides a default favicon, the water drop logo, but if you want to make your Developer Portal stand out, you should provide your own. The default Drupal favicon is 32 pixels high by 32 pixels wide. As a favicon is displayed only in the address bar and favorites (bookmarks) list, any favicon that you create should be similarly small.

---

## Procedure

To change the shortcut icon, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Appearance in the administrator dashboard.
3. Click Settings for the default theme.
4. Click Favicon in the Override Global Settings section.
5. Deselect the Use the favicon supplied by the theme check box.

6. Provide a path to a custom icon on the server by entering it in the Path to custom icon field. Alternatively, you can browse for, and upload, a favicon image under the Upload favicon image subheading.
7. Click Save configuration.  
Note: If the changes don't appear immediately in your browser, clear your browser's cache and reload the page. If you've bookmarked the site, you may need to delete the bookmark and then create it again, so the new favicon is used.

---

## Results

You have changed the favicon for the site.

---

## Changing the site email address

You can change the site email address in the Developer Portal.

---

## Before you begin

You must have administrator access to complete this task.

---

## Procedure

To change the site email address, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. Under the System heading, click Basic site settings.
4. In the SITE DETAILS section, enter the new site email address in the Email address field.
5. Click Save configuration.

---

## Results

You changed the email address for the site.

---

## Changing the site logo

You can change the site logo for your Developer Portal.

---

## Before you begin

You must have administrator access to complete this task.

---

## Procedure

To change the site logo, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Appearance in the administrator dashboard.
3. Click Settings for the default theme.
4. Click Logo image in the Override Global Settings section.
5. Deselect the Use the logo supplied by the theme check box.
6. Provide a path to your custom logo by completing the Path to custom logo field. Alternatively, you can browse and upload a logo image under the Upload logo image subheading.
7. Click Save configuration.  
Note: If the changes don't appear immediately in your browser, clear your browser's cache and reload the page.

---

## Results

You changed the site logo for your Developer Portal. How the site logo is used depends on the settings for your theme.

---

## Changing the site name or site slogan

You can change the site name, or the site slogan, for your Developer Portal.

---

## Before you begin

You must have administrator access to complete this task.

## About this task

---

The site name is used to identify the site and can be displayed in browsers. You can also set a site slogan for your theme.

## Procedure

---

To change the site name or site slogan, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration > System > Basic site settings in the administrator dashboard.
3. In the SITE DETAILS section, enter the new site name in the Site name field, or enter a new slogan in the Slogan field.
4. Click Save configuration.

## Results

---

The site name or site slogan is updated. How the site name and the site slogan are used depends on the settings for your site theme. For more information, see [Changing the visibility of site branding](#).

---

## Changing the visibility of site branding

You can toggle the visibility of your site branding elements for your Developer Portal.

## Before you begin

---

You must have administrator access to complete this task.

## About this task

---

The site branding block includes the site logo, name, and slogan. You can choose which of these branding elements you want to be displayed in this block.

## Procedure

---

To change the visibility of your site branding elements, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure > Block layout in the administrator dashboard.
3. Find the Site branding block in the Navigation section, and click Configure.
4. In the TOGGLE BRANDING ELEMENTS section, use the check boxes to select which branding elements you want to be visible.
5. Click Save block to save your changes.

## Results

---

You changed the visibility of your site branding elements. How these elements are used depends on the settings for your theme.

---

## Adding custom JavaScript to a custom theme

To increase customization of the Developer Portal, you can add custom JavaScript to custom themes that you install.

## Procedure

---

1. Add your JavaScript file to your theme. For example, in a sub-directory named js and the file `myfilename.js`.
2. Add the JavaScript file to your theme, by editing your `{custom_theme}.libraries.yml` file and adding in the JavaScript file.  
An example of the JavaScript file:

```
myfile:
 js:
 js/myfilename.js: {}
```

3. Add the following code to your theme's `.info.yml` file:

```
libraries:
 - {custom_theme}/myfile
```

- Note: You must replace `{custom_theme}` with your theme name. Also, both files must reference `myfile`.
4. Zip up the directory and upload it to your portal site. For more information, see [Installing additional themes](#).

---

## Applying a modified content type template

You can apply a modified content type template to override a default content type template in the Developer Portal. For example, you can override the product content type template to change its layout.

## Before you begin

---

You must have administrator access to complete this task.

You must also have a custom sub-theme, and a copy of an existing template that you want to update. For information about how to create a sub-theme, see [Creating a sub-theme](#).

To obtain a copy of an existing template, you can download a copy from the following locations on GitHub:

- [https://github.com/ibm-apiconnect/devportal/tree/master/ibm\\_apim/templates](https://github.com/ibm-apiconnect/devportal/tree/master/ibm_apim/templates)
- [https://github.com/ibm-apiconnect/devportal/tree/master/apic\\_api/templates](https://github.com/ibm-apiconnect/devportal/tree/master/apic_api/templates)
- [https://github.com/ibm-apiconnect/devportal/tree/master/apic\\_app/templates](https://github.com/ibm-apiconnect/devportal/tree/master/apic_app/templates)
- <https://github.com/ibm-apiconnect/devportal/tree/master/product/templates>
- <https://github.com/ibm-apiconnect/devportal/tree/master/consumerorg/templates>

Note that you must copy the template to your sub-theme, and then modify the content. Do not modify the content that is stored on GitHub.

## About this task

---

You can override the Developer Portal core templates by creating a custom sub-theme, and then adding your modified templates to the sub-theme folder. A sub-theme inherits all of the settings of its parent theme, apart from the settings that are overridden, such as by applying modified templates. After you have applied a modified content types template, these changes take precedence over the default content types template. The Developer Portal templates are formatted as Twig template files. The following levels are examples of where you can use templates:

- `html.html.twig` - the template for Developer Portal HTML pages
- `page.html.twig` - the body of Developer Portal HTML page
- `node.html.twig` - the template for all of the content nodes
- `comment.html.twig` - the template for all of the comment nodes
- `search-result.html.twig` - the template for search results
- `node--product.html.twig` - the template for nodes that use the Product content type
- `node--product--teaser.html.twig` - the template for the previews for the Product content type

For more information about Twig templates, see [Working with Twig templates](#) and [Debugging Twig templates](#) on the Drupal documentation website. Modifying Twig templates allows very fine-grained control over the structure of pages. However, it also means you might miss new features and defect fixes that are made to the original templates, as your Developer Portal is using your overrides instead of the originals. So, if you override a template, it is your responsibility to check the templates in the latest API Connect releases, and to ensure that any equivalent changes that are needed to your overrides to maintain functional equivalence are made.

Important:

Directly editing any API Connect theme files is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed.

## Procedure

---

To override a default content type template:

1. Apply the modifications that you require to the default template that you obtained.  
Warning: Although the ability to override templates is supported, if you override a template they are your responsibility. Templates can be from multiple sources and compatibility with an earlier version is not ensured. You must check the templates in the latest API Connect release to check whether you need to make equivalent changes to your overrides to maintain functional equivalence.
2. Create a directory in your custom sub-theme, and name it `templates`.
3. Add your modified template to the `templates` directory.  
Note: The modified template file name must be identical to the original name of the template file that it is overriding.
4. Log in to the Developer Portal UI as an administrator.
5. If the administrator dashboard is not displayed, click `Manage` to display it.
6. Install your custom sub-theme onto the Developer Portal by clicking `Appearance > Install new theme on the administrator dashboard`.
7. Click `Choose file`, select your sub-theme, and then click `Install`.
8. Click `Install newly added theme`, find your new theme in the list of `Uninstalled themes`, and click `Install` and set as default to set your new custom sub-theme as the default theme for your site.  
Your new theme, which contains the modified template file, is now set as the default theme, and is listed in the `Installed themes` section.

## Results

---

You successfully modified and applied a template to override a default content type template in the Developer Portal.

## Creating a sub-theme

---

Create a custom theme for your Developer Portal site by generating and configuring a sub-theme. A sub-theme is a theme that inherits all the resources of a specified parent theme. You can then override specific resources to configure your required customizations.

## Before you begin

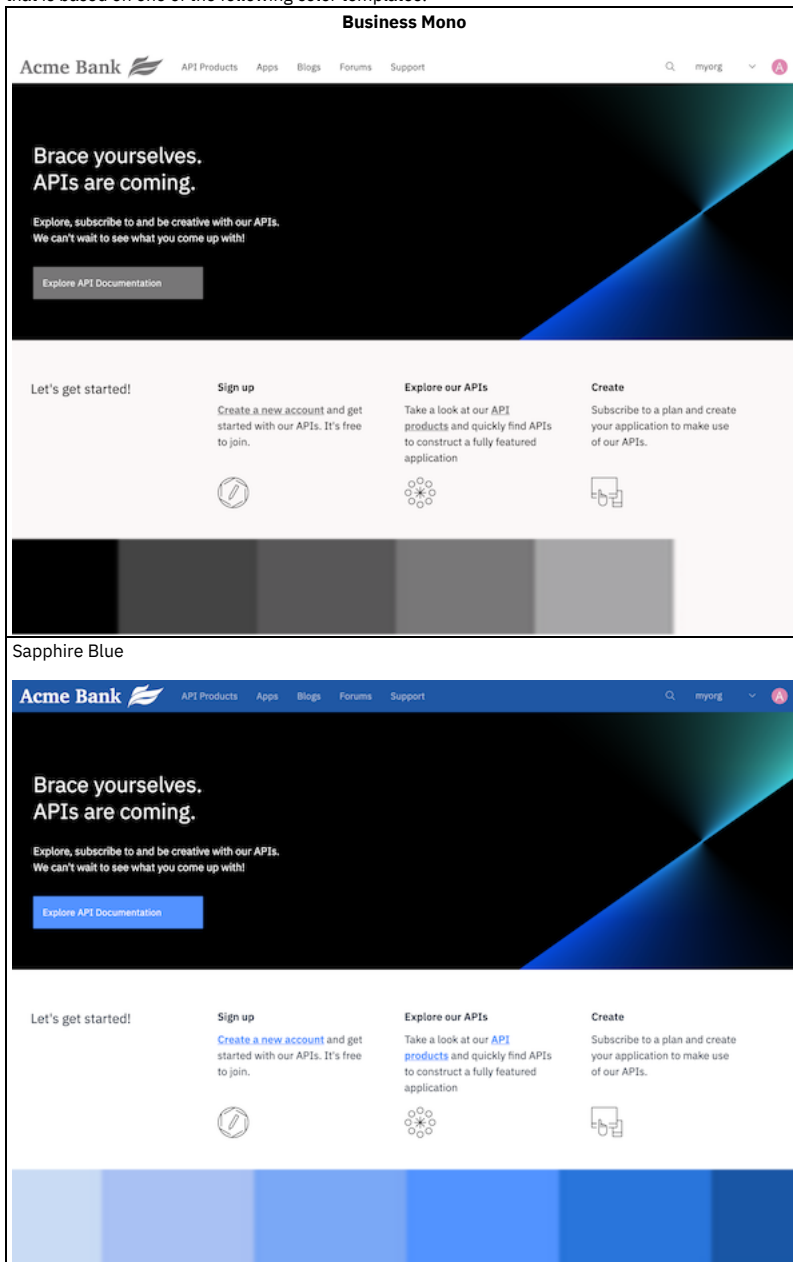
---

You must have administrator access to complete this task.



## About this task

Directly editing the API Connect theme isn't permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed. So, the way to create a custom theme is to create a custom sub-theme of the standard API Connect theme that the Developer Portal uses by default. You can also create a sub-theme that is based on one of the following color templates:



Emerald Green

The screenshot shows the Acme Bank website with an Emerald Green theme. The header includes the Acme Bank logo, navigation links for API Products, Apps, Blogs, Forums, and Support, a search icon, and a user profile icon labeled 'myorg'. The main banner features the headline 'Brace yourselves. APIs are coming.' and a sub-headline 'Explore, subscribe to and be creative with our APIs. We can't wait to see what you come up with!'. A green button labeled 'Explore API Documentation' is positioned below the text. The content area below the banner is divided into four columns: 'Let's get started!', 'Sign up' (with subtext 'Create a new account and get started with our APIs. It's free to join.' and a key icon), 'Explore our APIs' (with subtext 'Take a look at our API products and quickly find APIs to construct a fully featured application' and a network icon), and 'Create' (with subtext 'Subscribe to a plan and create your application to make use of our APIs.' and a document icon). A decorative bar at the bottom consists of five vertical segments in varying shades of green.

Golden Brown

The screenshot shows the Acme Bank website with a Golden Brown theme. The layout is identical to the Emerald Green version, featuring the same header, main banner with 'Brace yourselves. APIs are coming.' headline, and four-column content area. The 'Explore API Documentation' button and the bottom decorative bar are now in shades of brown and gold.

Ruby Red

The screenshot shows the Acme Bank website with a Ruby Red theme. The layout is identical to the other two versions, featuring the same header, main banner with 'Brace yourselves. APIs are coming.' headline, and four-column content area. The 'Explore API Documentation' button and the bottom decorative bar are now in shades of red and grey.

Selecting a color template provides you with a base theme that you can build on more easily to produce your required site branding and styling. Create a sub-theme by using the theme generator in the Developer Portal UI, customize the appearance as required, and then upload the compressed file back onto the site.

Important:

- You're not permitted to include any IBM® API Connect code within any custom themes that you create. Also, directly editing any API Connect themes on the file system is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed.
- All custom development is your responsibility. Although the use of custom themes is supported, IBM API Connect do not provide support in their development or modification.

For a guided example on creating a sub-theme, see [Tutorial: Creating a custom theme for the Developer Portal](#). In this example, you create a sub-theme, and then customize the styling by editing the `overrides.css` file.

## Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. On the administrator dashboard, click Appearance, > Generate sub-theme.  
The Generate sub-theme page is displayed.
3. Enter a Sub-theme name, and select the Sub-theme type that you would like to use. The name can contain only lowercase characters a - z, and numerals 1 - 9.
  - CSS - Cascading Style Sheets; the language that is used to describe how to display the HTML elements in the theme.
  - SCSS - Sass CSS; a superset of CSS. SCSS contains all the features of CSS, but is extended to include the features of Sass as well. However, SCSS must be compiled into CSS before it can be installed and used in the Developer Portal.
4. Select the template to base your sub-theme on:
  - Default - the standard connect\_theme that the Developer Portal is based on.
  - Business Mono - a black and white color theme.
  - Sapphire Blue - a blue color theme.
  - Emerald Green - a green color theme.
  - Golden Brown - a brown color theme.
  - Ruby Red - a red color theme.
5. Click Generate.  
Your sub-theme is available to download from the Generate sub-theme page.
6. Click your new sub-theme and save it to your preferred location.
7. Extract the resources from the sub-theme file, and then edit the specific resources to configure the customizations that you require.  
Define your theme by creating metadata in the `your_theme_name.info.yml` file. You can then customize the appearance of your theme by configuring the `overrides.css` file. For detailed information about how to customize themes, see [Theming Drupal](#).
8. After you have completed your customizations, compress the sub-theme resources back into the .zip file.  
Note that the following file extensions are also supported: tar, `tgz`, `gz`, and `bz2`.
9. Install your new sub-theme onto the Developer Portal by clicking Appearance, > Install new theme.
10. Click Choose file, select your sub-theme, and then click Install.
11. Click Install newly added theme, find your new theme in the list of Uninstalled themes, and click Install and set as default to set your new custom sub-theme as the default theme for your site.  
Your new theme is now set as the default theme, and is listed in the Installed themes section.

## Results

---

You created and installed a new sub-theme for your Developer Portal site.

## What to do next

---

You can edit the display settings for your new theme, by clicking Settings next to your theme.

## Related information

---

- [Tutorial: Creating a custom theme for the Developer Portal](#)

## Installing additional themes

---

You can install additional themes in the Developer Portal.

## Before you begin

---

You must have administrator access to complete this task.

## Procedure

---

To install additional themes, complete the following steps:

1. Click Appearance in the administrator dashboard.
2. Click + Install new theme.
3. You can enter a path to the theme in the Install from a URL field. Alternatively, you can upload a theme under the Upload a module or theme archive to install heading.
4. Click Install.
5. Click Enable newly added themes.

6. Click Enable and set default for the theme that you want to be displayed in the Developer Portal. Your theme is displayed in the Developer Portal when you navigate to the home page.

---

## Deleting themes

You can delete a theme from your Developer Portal site.

### Before you begin

---

You must have administrator access to complete this task.

### About this task

---

If you have a custom theme which is disabled, you can remove it from the Developer Portal. Deleting a theme is useful if you have identified a theme that you do not want to use again.

### Procedure

---

1. Click Appearance from the administrator dashboard.
2. Identify the custom theme that you want to delete, and ensure that it is disabled.
3. Click Delete against the theme.

The theme is now deleted from your Developer Portal site.

### Results

---

You deleted a disabled theme in the Developer Portal.

---

## Content

Control multiple content entities in the Developer Portal, including the customization and restriction of specific content.

A *content entity* is an item of content data, which can consist of text, HTML markup, images, attached files, and other data that is intended to be displayed to site visitors. Content entities can be defined by the core software or by modules. Content entities are grouped into *entity types*, which have different purposes and are displayed in different ways on the site. Most entity types are also divided into *entity sub-types*, which are divisions within an entity type to allow for smaller variations in how the entities are used and displayed.

The Developer Portal site contains the following content entity types:

- API
- Application
- Article
- Basic page
- Blog post
- Book page
- Consumer Organization
- FAQ
- Forum Topic
- Product

Within content entity items the data is stored in individual *fields*, each of which holds one type of data, such as formatted or plain text, images or other files, or dates. Field types can be defined by the core software or by modules.

Fields can be added by an administrator on content entity sub-types, so that all the entity items of a given entity sub-type have the same collection of fields available. When you create or edit entity items, you are specifying the values for the fields on the entity item.

To learn how to configure the content entities of your Developer Portal site, use the following topic links.

- [Adding content elements in the Developer Portal](#)  
You can add content elements in the Developer Portal.
- [Configuring and restricting content in the Developer Portal](#)  
You can configure and restrict certain content elements in the Developer Portal.
- [Turning content on or off in the Developer Portal](#)  
You can turn certain content elements on or off in the Developer Portal

---

## Adding content elements in the Developer Portal

You can add content elements in the Developer Portal.

To learn how to add various content elements in the Developer Portal, use the following topic links:

- [Adding and configuring meta tags](#)  
You can customize the Developer Portal by adding and configuring meta tags.
- [Adding custom pages](#)  
You can create custom pages within the Developer Portal. You can add a basic page for static content, or an article for news content.
- [Integrating Twitter data into the social block](#)  
You can retrieve and integrate data from a Twitter account of interest and display it into a social block in a Developer Portal site.
- [Adding content in multiple languages](#)  
You can create content in multiple languages in the Developer Portal.
- [Adding custom pages to APIs and Products](#)  
After you create a custom page in the Developer Portal, you can add a link to the new page to any APIs and Products that exist in the Developer Portal.
- [Adding new frequently asked questions](#)  
You can add new FAQs to the Developer Portal.
- [Applying an image to an API or Product](#)  
You can apply an image to an API or a Product in the Developer Portal.
- [Attaching a documentation file to APIs](#)  
You can attach a file to APIs in the Developer Portal, and display them as documentation attachments. You can attach multiple files to APIs.
- [Configuring blogs](#)  
You can post new blog entries in the Developer Portal, and configure them to suit your requirements. You can also turn blogs off.
- [Configuring the taxonomy menu block](#)  
To display your tag hierarchy in the Developer Portal, you must configure the taxonomy menu block.
- [Customizing the URL alias for a specific API or Product page](#)  
You can customize the URL of an API or Product page from the default alias that they are assigned.
- [Embedding multimedia in site content](#)  
You can embed multimedia elements in site content in the Developer Portal.
- [Linking from one piece of site content to another](#)  
You can link from one piece of site content to another in the Developer Portal.
- [Linking to social media sites](#)  
You can link to social media sites from the Developer Portal. You can configure where and how these links are displayed and which sites you want to link to.
- [Managing tags in the Developer Portal](#)  
These instructions show you how to add new tags to your taxonomy, and how to manage the tag hierarchy in the Developer Portal. You can use tags to classify your Developer Portal content.
- [Posting a poll](#)  
You can post a poll in the Developer Portal.
- [Adding images to site content](#)  
You can add .jpg, .png and .gif images to your site content in the Developer Portal.

---

## Adding and configuring meta tags

You can customize the Developer Portal by adding and configuring meta tags.

### Before you begin

---

You must have administrator access to complete this task.

### About this task

---

Meta tags provide preview information that is displayed for a link to a website. The preview information can include images, descriptions, tags, and links. Meta tags are used when you link to external website, for example Facebook, Twitter, and Slack.

You can see which entities have the meta tag form available on their respective edit pages by default, and can be configured, Configuration\_>Search and metadata\_>Metatags\_>Settings:

- CONTENT
  - API
  - APPLICATION
  - ARTICLE
  - BLOG POST
  - BOOK PAGE
  - CONSUMER ORGANIZATION
  - FAQ
  - FORUM TOPIC
  - BASIC PAGE
  - PRODUCT
- TAXONOMY TERM
  - FORUMS
  - TAGS
- USER
  - USER

### Procedure

---

1. In the administrator dashboard, click Extend\_>Update, then click List.
2. Enter `meta tag` in the Filter list field.
3. Select the check boxes for the Meta tags you want to enable, click Enable.

- [Disabling external search engine indexing](#)

If you do not want your Developer Portal site content to be indexed by external search engines, then you can disable the indexing.

---

## Disabling external search engine indexing

If you do not want your Developer Portal site content to be indexed by external search engines, then you can disable the indexing.

---

### Before you begin

You must have administrator or content author access to complete this task.

---

### About this task

By disabling the external search indexing, you can increase the confidentiality of your content and restrict its exposure.

---

### Procedure

1. If the administrator dashboard is not displayed, click **Manage** to display it.
2. On the administrator dashboard, click **Configuration**.
3. In the **SEARCH AND METADATA** section, click **Metatag**.
4. Click **Edit** for the **GLOBAL** type.
5. Expand the **ADVANCED** section and, in the **Robots** section, select **Prevents search engines from indexing this page**.
6. Click **Save**.

---

### Results

You have disabled external search engine indexing.

---

## Adding custom pages

You can create custom pages within the Developer Portal. You can add a basic page for static content, or an article for news content.

---

### Before you begin

You must have administrator or content author access to complete this task.

---

### About this task

Use basic pages for your static content, such as an "About us" page. Typically, a basic page is used for static content that can be linked into the main navigation bar, although this is not a requirement. Use articles for time-sensitive content, like news, press releases, or blog posts.

---

### Procedure

To add custom pages, complete the following steps.

1. Log in to the Developer Portal as an administrator or content author.
2. If the administrator dashboard is not displayed, click **Manage** to display it.
3. In the administrator dashboard, click **Content**.
4. Click **+Add content**, then click either **Basic page** or **Article**, depending on the type of content you want to display.
5. Enter a page title, select the language, and enter the content.
6. Specify the remaining page options as required.  
If you want your new page to be linked to from a menu, you must complete the **Menu settings** section.
7. Optional: Click **Edit summary** and specify a summary page description.
8. Click **Save** to publish the new page immediately. If you don't want to publish the new page immediately, ensure that you **deselect Published** before saving.

---

### Results

You added a new custom page to the Developer Portal. You can edit the page after it is created by clicking **Edit** for the custom page.

---

## Integrating Twitter data into the social block

You can retrieve and integrate data from a Twitter account of interest and display it into a social block in a Developer Portal site.

---

### Before you begin

You must have administrator access to complete this task.

You must have a Twitter App for the Twitter account of interest. The Twitter App contains information that is used to configure the social block. For information on creating Twitter Apps, see <https://apps.twitter.com/>.

## Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. On the administrator dashboard, click Configuration.
3. In the WEB SERVICES section, click IBM Social Block settings.
4. Using the information from your Twitter App, enter the Consumer Key, Consumer Secret, Access Token, and Token Secret in the corresponding fields.
5. Click Save configuration.

Twitter data from a Twitter account of interest has been integrated with the Developer Portal, and is displayed on the social block of the site.

- **Editing the social block**

After you have integrated Twitter data into the Developer Portal social block, you can edit the content, appearance, and functionality of the social block.

---

## Editing the social block

After you have integrated Twitter data into the Developer Portal social block, you can edit the content, appearance, and functionality of the social block.

## Before you begin

---

You must have administrator access to complete this task.

You must have Twitter data integrated into the Developer Portal social block. For more information, see [Integrating Twitter data into the social block](#).

## Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. On the administrator dashboard, click Structure > Pages.
3. Click Edit next to the Welcome page (or next to the page that contains the social block that you want to edit).
4. From the Variants navigation tree, select Panels > Content, and click Edit next to the Social Block.

The Edit block window is displayed.

5. Edit the social block by configuring any of the following options:

Title

Update the title as required.

Number of tiles to display

Configure the number of tiles that you want to display on the social block by entering a number in the Number of tiles to display field. The default number of tiles that are displayed on the social block is 9, and there is a limit of 100 tiles.

FORUM

Use the check boxes to specify the forum topics that you want to display.

Get tweets from

From the Get tweets from drop-down list, select whether you want to display Twitter data from users or a search term.

Twitter search parameter

Enter the Twitter search parameter.

Types of tweets to display

Select the types of tweets that you want to display from the drop-down list. You can select to display tweets, or tweets and replies.

6. After you have configured the social block, click Update block.

Your social block is configured, and the configurations are implemented when you return to the home page.

---

## Adding content in multiple languages

You can create content in multiple languages in the Developer Portal.

## Before you begin

---

You must have administrator or content author access to complete this task.

## About this task

---

You can provide custom pages in multiple languages to your APIs and Products to improve your customer experience and understanding. By default, multiple languages are enabled in the Developer Portal.

Note: In API Designer, translations for APIs and products must be done within a document. For more information, see [Using x-ibm-languages to create multilingual API and Product documentation](#).

## Procedure

---

1. Log in to the Developer Portal as an administrator or content author.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. On the administrator dashboard, click Content.

4. If you are creating a new page to be translated, click +Add content.
  - a. Click Basic page, then select type, for example **Basic page**.
  - b. Enter a title for your custom page in the Title field.
  - c. Enter your information in the Body field.
  - d. Click Save.
  - e. Click Translate.
  - f. Click Add for the language you want to translate.
  - g. Provide the language content.
  - h. Click Save (this translation).
5. If you are translating an existing page, find the page that you want to update in the list.
  - a. Click Translate from the drop-down list for the page.
  - b. Click Add for the language you want to translate.
  - c. Provide the language content.
  - d. Click Save (this translation).

---

## Results

You successfully created a translated version of your content.

---

## Adding custom pages to APIs and Products

After you create a custom page in the Developer Portal, you can add a link to the new page to any APIs and Products that exist in the Developer Portal.

---

### Before you begin

You must have administrator or content author access to complete this task.

You must have an API or Product to add the custom page to.

---

### About this task

You can add custom pages to your APIs and Products to provide further information on their use and value.

For example, you can create a custom documentation page on the implementation of an API in a specific circumstance, and add this custom documentation page to the API.

---

### Procedure

1. Log in to the Developer Portal as an administrator or content author.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. On the administrator dashboard, click Content, then click +Add content.
4. Click Basic page.

The Create Basic page window is displayed.
5. Enter a title for your custom page in the Title field.
6. Enter your information of interest in the Body field.
7. If you want to add the custom page to a specific Product, enter the Product name in the Link to one or more specific Products field. As you type into this field, a list of available Products is displayed for you to select. You can add more Products by clicking Add another item. If you want to add the custom page to all Products, select the Link to all Products check box.
8. If you want to add the custom page to a specific API, select the APIs tab, and then enter the API name in the Link to one or more specific APIs field. As you type into this field, a list of available APIs is displayed for you to select. You can add more APIs by clicking Add another item. If you want to add the custom page to all APIs, select the Link to all APIs check box.
9. Optional: Configure the additional settings of the custom page to your specification.
10. Click Save.

---

### Results

Your custom page is created and linked to the APIs or Products that you have specified. You can edit the page after it is created, and update the APIs and Products that it is linked to, by clicking Edit for the custom page.

---

## Adding new frequently asked questions

You can add new FAQs to the Developer Portal.

---

### Before you begin

You must have administrator or content author access to complete this task.

---

### About this task



To add new FAQs, you create a new FAQ node and add questions and answers to that node. Each FAQ node represents an FAQ topic, and each topic can contain multiple questions and answers.

## Procedure

---

To add new FAQs, complete the following steps:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click **Manage** to display it.
3. Click **Content** in the administrator dashboard.
4. Click **+Add content**.
5. Click **FAQ**.  
The **Create FAQ** page displays.
6. Provide a **Title** for the new FAQ topic.
7. Enter the question in the **Question** field, and enter the answer in the **Answer** field.
8. Optional: If you have more content for this FAQ topic, click **Add another item** and complete the **Question** and **Answer** fields. Repeat this step as required.
9. Click **Save** to save your new FAQ topic.  
Your new FAQ topic is displayed in the **Support** section of the Developer Portal.

## Results

---

You successfully added new FAQs to the Developer Portal.

## What to do next

---

You can edit the current content of FAQs by clicking **Content** in the administrator dashboard, finding the FAQ content that you want to edit, and clicking **Edit** against that FAQ.

## Applying an image to an API or Product

---

You can apply an image to an API or a Product in the Developer Portal.

## Before you begin

---

You must have administrator or content author access to complete this task.

## Procedure

---

To add an image to an API or Product, complete the following steps:

1. If the administrator dashboard is not displayed, click **Manage** to display it.
2. Click **Content** in the administrator dashboard.
3. Click **Edit** next to the API or Product that you want to edit.
4. Under the **Image** heading, click **Choose File** to select an image.
5. Click **Save and keep published**, to publish the updated content, or select **Save and unpublish** if you want to publish the updated content later.  
You have added a new picture to the selected API or Product.

## Attaching a documentation file to APIs

---

You can attach a file to APIs in the Developer Portal, and display them as documentation attachments. You can attach multiple files to APIs.

## Before you begin

---

You must have administrator or content author access to complete this task.

## About this task

---

You can attach only one file at a time. The default file upload size limit applies to the files that you want to attach, and you can attach up to 10 files to an API by default.

Note: You can change the default limit for the number of files that you want to attach.

An administrator can configure the types of file that are uploaded, and the number of files that a user can upload.

The following file types can be uploaded:

- txt doc pdf
- xls ppt pptx
- docx xlsx rtf
- odt ods odp md json
- yaml tgz tar zip

## Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. Click Edit next to the API you want to attach the file to.
4. In the Documentation section, click Browse and identify the file that you want to upload, then click Open.
5. Click Upload, then click Save.
6. Optional: Clear the Display checkbox for the file if you want to upload and attach the file to the API, but not show it in the Developer Portal.
7. Optional: Enter a description for your file in the Description field. This description can be used as the label of the link to the file.
8. Optional: If you uploaded multiple files, you can use the drag handle next to the file name to rearrange them.

## Results

---

You successfully attached a documentation file to your API.

## Configuring blogs

You can post new blog entries in the Developer Portal, and configure them to suit your requirements. You can also turn blogs off.

## Before you begin

---

You must have administrator or content author access to complete this task.

## Procedure

---

To post a new blog entry, complete the following steps:

- If the administrator dashboard is not displayed, click Manage to display it.
- Click Content in the administrator dashboard.
- Click Add content.
- Click Blog post.
- Fill in the Title and Body fields.
- Click Save and keep published, to publish the updated content, or select Save and unpublish if you want to publish the updated content later.  
You have posted a new blog entry.

To configure a blog post, complete the following steps:

- If the administrator dashboard is not displayed, click Manage to display it.
- Click Content in the administrator dashboard.
- Click Edit next to the blog post that you want to configure.
- You can configure the selected blog from this view.
- Click Save and keep published, to publish the updated content, or select Save and unpublish if you want to publish the updated content later.  
You have configured the selected blog post.

## Configuring the taxonomy menu block

To display your tag hierarchy in the Developer Portal, you must configure the taxonomy menu block.

## Before you begin

---

You must have administrator or content author access to complete this task.

## About this task

---

Along with controlling how your tag hierarchy is displayed through configuring your taxonomy menu block, you can configure options for the taxonomy tree. The following options can be configured for your taxonomy menu block:

- Vocabulary
- Parent option
- Depth
- Node options and content
- Region settings
- Visibility settings

Important: The taxonomy block will appear after configuration only if there are tags for it to display.

## Procedure

---

1. In the Administrator dashboard, click Structure > Blocks.
2. Click configure for the Taxonomy Menu Block (Tags) block.
3. Configure the options that are available for your taxonomy menu block.
4. When you have configured your taxonomy menu block, click Save block.  
You have configured your taxonomy menu block, and your tag hierarchy is displayed based on your configurations.

---

## Customizing the URL alias for a specific API or Product page

You can customize the URL of an API or Product page from the default alias that they are assigned.

### Before you begin

---

You must have administrator or content author access to complete this task.

### About this task

---

Creating a custom URL, or path, alias is useful if you want to make the URL relevant to the API or Product page.

### Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. On the administrator dashboard, click Content.
3. Identify the API or Product page that you want to customize the URL alias for, and click Edit.
4. In the Path Alias text field, enter the custom URL alias that you want to assign your API or Product page.  
Note: All Path Alias must be unique, and can contain only the following characters: [A-Za-z0-9\_-].
5. Click Save to publish the updated content.

### Results

---

You have applied a custom URL alias for your specified API or Product page, and it is displayed in the URL of your browser when the API or Product page is selected. Depending on whether it is added to an API or a Product page, the application of the Path Alias differs. For example, if you add **my\_new\_path\_alias**, you will have:

- `<my_site_url>/api/my_new_path_alias` for an API
- `<my_site_url>/product/my_new_path_alias` for a Product

---

## Embedding multimedia in site content

You can embed multimedia elements in site content in the Developer Portal.

### Before you begin


---

You must have administrator or content author access to complete this task.

### Procedure

---

To embed multimedia in site content, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. Under the TITLE heading, click the title of the required content.
4. Click the Edit tab.
5. Use the Insert Media Embed icon  in the in the WYSIWYG editor to embed multimedia in the content.  
Note: The Insert Media Embed icon is available only if you have selected Full HTML mode.
6. Click Save and keep published to publish the new page immediately, or select Save and unpublish if you want to publish the page later.  
You have embedded multimedia in the selected site content.

---

## Linking from one piece of site content to another

You can link from one piece of site content to another in the Developer Portal.

### Before you begin

---

You must have administrator or content author access to complete this task.

### Procedure

---

To link from one piece of site content to another, complete the following steps:

1. Click Content in the administrator dashboard.
2. Under the TITLE heading, click the title of the required content.
3. Select the Edit tab.



4. Use the Link to content icon in the WYSIWYG editor.
5. Click Save.  
You have linked the selected site content to another site content element.

---

## Linking to social media sites

You can link to social media sites from the Developer Portal. You can configure where and how these links are displayed and which sites you want to link to.

---

### Before you begin

You must have administrator access to complete this task.

---

### Procedure

1. Click Structure in the administrator dashboard.
2. Click Blocks.
3. Under the Disabled heading, locate the Follow site block.
4. Select the location for your social media block from the drop-down list for Follow Site in the REGION column.  
The Follow Site block will appear under the heading that corresponds to the region you selected in the REGION drop-down list.
5. Required: Click Save blocks before you begin configuring your new block.
6. Click configure for the Follow Site block.
7. In the Block title field, type the name of your block.
8. Under the Default block title heading, select one of the following leading text options for your block using the radio buttons for each option:
  - "Follow *your.portal.site* on"
  - "Follow me on"
  - "Follow us on"
9. Use the User pages check box to decide whether you want your block to display on your user's profile pages.
10. Use the Alignment and Icon Style drop-down lists to adjust the appearance of your block.
11. Click Save block.
12. Click Configuration in the administrator dashboard.
13. Under the WEB SERVICES heading, click Follow.
14. Type the URL under the URL heading for each of the social media sites you want your users to access. The following is an example of a link to the @ibmapiconnect twitter handle:  
`https://twitter.com/ibmapiconnect`
15. Click Submit.

---

## Managing tags in the Developer Portal

These instructions show you how to add new tags to your taxonomy, and how to manage the tag hierarchy in the Developer Portal. You can use tags to classify your Developer Portal content.

---

### Before you begin

You must have administrator or content author access to complete this task.

---

### Procedure

To manage your tags, complete the following steps

1. If the administrator dashboard is not displayed, click Manage to display it.
  2. Click Structure > Taxonomy > Tags in the administrator dashboard.  
The List window for Tags is displayed.
  3. You can add new terms, delete, and reorganize terms from this view.
    - a. To add a new term, click Add term, define the new term, then click Save.
    - b. To delete a term, select Delete from the OPERATIONS menu next to the term that you want to delete. Then click Delete to confirm.
    - c. To reorganize terms, ensure that the row weights are hidden by clicking Hide row weights if necessary, then use the drag-and-drop handles to manage the tag hierarchy. Terms can be dragged to organize a parent grouping hierarchy. Then click Save.
- [Tagging APIs based on their lifecycle phase](#)  
In the Developer Portal, you can have APIs automatically tagged with their lifecycle phase; realized, identified, or specified.

---

### Related tasks

- [Editing tags for a specific item](#)

---

## Tagging APIs based on their lifecycle phase

In the Developer Portal, you can have APIs automatically tagged with their lifecycle phase; realized, identified, or specified.

## Before you begin

---

You must have administrator or content author access to complete this task.

## About this task

---

APIs are tagged in the API Manager with a lifecycle phase as follows:

- Realized (default) - The API is in the implementation phase.
- Identified - The API is in the early conceptual phase and is neither fully designed nor implemented.
- Specified - The API has been fully designed and passed an internal milestone, but has not yet been implemented.

You can configure the Developer Portal to tag APIs with their lifecycle phase automatically.

## Procedure

---

Tagging APIs with their phase is disabled by default. To enable the function:

1. If the administrator dashboard is not displayed, click **Manage** to display it.
2. On the administrator dashboard, click **Configuration**.
3. In the **SYSTEM** section, click **IBM API Connect**.
4. Select **Automatically tag APIs with their phase**, and then click **Save configuration**.

## What to do next

---

You can manage the tag hierarchy of your APIs. For more information, see [Managing tags in the Developer Portal](#).

## Posting a poll

You can post a poll in the Developer Portal.

## Before you begin

---

You must have administrator or content author access to complete this task.

## Procedure

---

To post a poll, complete the following steps:

1. Click **Content** in the administrator dashboard.
2. Click **Add content**.
3. Click **Poll**.
4. Enter the poll question in the **Poll question** field.
5. Under the **Choice** heading, add answers to the available fields as choices.
6. From the **Poll duration** drop-down menu, select a time limit for the poll.
7. Click **Save**.

You have posted a poll.

## What to do next

---

See [Editing sample content](#) to learn how to add a links to your poll to sample content.

## Adding images to site content

You can add .jpg, .png and .gif images to your site content in the Developer Portal.

## Before you begin

---

You must have administrator or content author access to complete this task.

## Procedure

---

To upload images for use in site content, complete the following steps:

1. If the administrator dashboard is not displayed, click **Manage** to display it.
2. Click **Content** in the administrator dashboard.
3. Under the **TITLE** heading, click the title of the required content.
4. Click the **EDIT** tab.



5. Use the Image icon in the in the WYSIWYG editor to add an image.
6. Click Save and keep published to publish the new page immediately, or select Save and unpublish if you want to publish the page later.

---

## Configuring and restricting content in the Developer Portal

You can configure and restrict certain content elements in the Developer Portal.

To learn how to configure and restrict content in the Developer Portal, use the following topic links:

- [Creating content types](#)  
Content types define default fields for editors to add content on your Developer Portal site and are the building blocks for authoring content. You can control which content types are available
- [Configuring documentation upload limitations](#)  
You can attach documentation to an API by uploading it in the Developer Portal. You can configure the API content type to place limitations on the uploaded documentation.
- [Configuring image upload limitations](#)  
You can attach an image file to an API by uploading it in the Developer Portal. You can configure the API content type to place limitations on the uploaded image files.
- [Configuring the appearance of ratings in the Developer Portal](#)  
You can configure the appearance of ratings in the Developer Portal.
- [Customizing the privacy policy statement](#)  
You can define and customize a privacy policy statement that sets out the privacy conditions of your Developer Portal site. The privacy policy can be linked to from the cookie compliance banner.
- [Customizing the terms of use statement](#)  
You can define and customize a terms of use statement that sets out the terms and conditions that your users must accept to use your Developer Portal site.
- [Editing a site comment](#)  
You can edit site comments in the Developer Portal.
- [Editing sample content](#)  
You can edit sample content, such as basic pages, in the Developer Portal.
- [Editing tags for a specific item](#)  
You can edit tags for a specific item in the Developer Portal.

---

## Creating content types

Content types define default fields for editors to add content on your Developer Portal site and are the building blocks for authoring content. You can control which content types are available

### Before you begin

---

You must have administrator access to complete this task.

### About this task

---

After you create your content type, it is added to the following list of default content types that are present in the Developer Portal:

- API
- Application
- Article
- Basic page
- Blog post
- Book page
- Consumer organization
- Forum topic
- Product

The content type that you create, and the default content types, can be edited and configured further to your specifications.

### Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Content types.
4. Click +Add content type.
5. Enter a name for your content type in the Name field.
6. Optional: Enter the text that you want to appear on the Add new content page, in the Description field.
7. Enter a title field label for your content type in the Title field label field.
8. Optional: Configure any of the following options for your content type:
  - Submission form settings
  - Publishing options
  - Language settings
  - Display settings

- Menu settings
9. After you have configured your content type to your specifications, click Save and manage fields.  
You have created a new content type.

---

## What to do next

You can add fields to your new content type; see [Adding fields to content types](#).

- [Adding fields to content types](#)  
Fields can be added to content types in the Developer Portal for customization, and to add functionality.

---

## Adding fields to content types

Fields can be added to content types in the Developer Portal for customization, and to add functionality.

---

## Before you begin

You must have administrator access to complete this task.

---

## About this task

You can specify the type of data that a field can store, and the form element to edit the data with. The type of form element that is available is dependent on the type of data that your field stores.

Note: You can specify the number of values that a field can store, including an unlimited amount. You can configure the value after the field is created. Reducing the number of values that can be stored after the field is created might lead to data loss.

---

## Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Content types, and identify the content type that you want to add a field to.
4. Click Manage fields for the content type, then click Add field.
5. Select a field type or select an existing field to reuse, then enter a label for your field in the Label field.  
The label that you specify for your field is displayed in the UI.
6. Optional: If you want to edit the machine name that is automatically created based on your label, click Edit and enter the new name.  
You cannot change the machine name after the field is created.
7. Click Save and continue.
8. Modify the field type specific settings as required, then click Save field settings.  
You have added a field to your content type.

---

## Configuring documentation upload limitations

You can attach documentation to an API by uploading it in the Developer Portal. You can configure the API content type to place limitations on the uploaded documentation.

---

## Before you begin

You must have administrator access to complete this task.

---

## About this task

The set of configurable limitations that you can place on uploading documentation is different to the set of limitations that you can place on uploading images.

The file size limit that is applied when you upload documentation is also different to the limit that is applied when you upload images.

---

## Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Content types.
4. For the API content type, click Manage fields.
5. For the Documentation field, click Edit.
6. Configure the type of files that are allowed to be uploaded in the Allowed file extensions field.
7. Enter the file size upload limit in the Maximum upload size field.  
The default size is 10MB, while the maximum upload size possible is 64MB.
8. Click Save settings.  
You have configured the documentation upload limitations.

---

## Configuring image upload limitations

You can attach an image file to an API by uploading it in the Developer Portal. You can configure the API content type to place limitations on the uploaded image files.

### Before you begin

---

You must have administrator access to complete this task.

### Procedure

---

1. If the administrator dashboard is not displayed, click **Manage** to display it.
2. Click **Structure** in the administrator dashboard.
3. Click **Content types**.
4. For the API content type, click **Manage fields**.
5. For the **Image** field, click **Edit**.
6. Configure the type of image files that are allowed to be uploaded in the **Allowed file extensions** field.
7. Specify the maximum and minimum allowed image sizes in the **Maximum image resolution** and **Minimum image resolution** fields.
8. Enter the file size upload limit in the **Maximum upload size** field.  
The default size is 2MB, while the maximum upload size possible is 64MB.
9. Click **Save settings**.  
You have configured the image upload limitations.

---

## Configuring the appearance of ratings in the Developer Portal

You can configure the appearance of ratings in the Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

### About this task

---

You can configure the appearance of ratings for only the following content types in the Developer Portal:

- APIs
- Applications
- Products

Note: The Applications and Products content types rating functionality is set to hidden by default.

### Procedure


---

These following steps describe how to customize the rating on a selected API, Application, or Product. To ensure that the configured rating is also displayed in the list view of APIs, Applications, or Products, then you must repeat these steps on the **Teaser** tab in the **Manage Display** window:

1. If the administrator dashboard is not displayed, click **Manage** to display it.
2. Click **Structure** in the administrator dashboard.
3. Click **Content types**.
4. Select **Manage display** from the list in **OPERATIONS** column, next to the required content type.





5. Click the Settings icon , then select the style you require, then click Update.

### Format settings: Voting api formatter

#### Styles

Bootstrap stars ▼

Readonly

Show results

Show own vote

Show own cast vote instead of results. (Useful on add/ edit forms with rate widget).

**Update** **Cancel**

6. Click Save. You have configured the appearance of the ratings feature for the required content type.

---

## Customizing the privacy policy statement

You can define and customize a privacy policy statement that sets out the privacy conditions of your Developer Portal site. The privacy policy can be linked to from the cookie compliance banner.

### Before you begin

You must have administrator access to complete this task.

### Procedure

To define or customize a privacy policy, complete the following steps:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Content, then enter `privacy` into the title search filter, and click Filter.  
The Privacy Policy content item is displayed in the list of content.
4. Click Edit next to the Privacy Policy content item.
5. Update the content as required. For example, you can complete the following tasks:
  - a. Edit the text of the privacy policy statement.
  - b. Set the display settings.
  - c. Link the content to Products and APIs.
  - d. Enter an explanation of the changes that have been made to the privacy policy since the last version.
6. Click Save to save your changes.

### Results

The privacy policy statement is successfully updated.

### What to do next

You can set your cookie compliance banner to link to your updated privacy policy. For more information, see [Enabling a cookie compliance banner](#).

### Related tasks

- [Enabling a cookie compliance banner](#)
- [Customizing the terms of use statement](#)

---

## Customizing the terms of use statement

You can define and customize a terms of use statement that sets out the terms and conditions that your users must accept to use your Developer Portal site.

## Before you begin

---

You must have administrator access to complete this task.

## Procedure

---

To define or customize a terms of use statement, complete the following steps:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Content, then enter `terms` into the title search filter, and click Filter.  
The Terms of use content item is displayed in the list of content.
4. Click Edit next to the Terms of use content item.
5. Update the content as required. For example, you can complete the following tasks:
  - a. Edit the text of the terms of use statement.
  - b. Set the display settings.
  - c. Link the content to Products and APIs.
  - d. Enter an explanation of the changes that have been made to the terms of use since the last version.
6. Click Save to save your changes.

## Results

---

The terms of use statement is successfully updated.

## What to do next

---

You can force new users to accept the terms and conditions on the Developer Portal. For more information, see [Forcing new users to accept terms and conditions](#).

## Related tasks

---

- [Enabling a cookie compliance banner](#)
- [Customizing the privacy policy statement](#)

## Editing a site comment

---

You can edit site comments in the Developer Portal.

## Before you begin

---

You must have administrator or content author access to complete this task.

## Procedure

---

To edit a site comment, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. Click the Comments tab.
4. In the OPERATIONS column, click Edit alongside the comment that you want to edit.
5. Edit the content of the comment as required.
6. Click Save.  
You have edited the selected comment.

## Editing sample content

---

You can edit sample content, such as basic pages, in the Developer Portal.

## Before you begin

---

You must have administrator or content author access to complete this task.

## Procedure

---

To edit sample content, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. In the OPERATIONS column, click Edit alongside the content item that you want to edit.
4. Edit the sample content of the selected element as required.

5. Click Save and keep published, to publish the updated content, or select Save and unpublish if you want to publish the updated content later.

---

## Editing tags for a specific item

You can edit tags for a specific item in the Developer Portal.

### Before you begin

---

You must have administrator or content author access to complete this task.

You must have some tags that are available to apply to your content; see [Managing tags in the Developer Portal](#).

### Procedure

---

To edit tags for a specific item, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. In the list of content items, click Edit alongside the content item that you want to update.
4. In the Tags section, select the required tag.
5. To add another tag, click Add another item, then select the required tag.
6. If you want to publish the updated content, ensure the Published check box is selected. Deselect the check box if you want to publish the updated content later.
7. Click Save.

### Related tasks

---

- [Managing tags in the Developer Portal](#)

---

## Turning content on or off in the Developer Portal

You can turn certain content elements on or off in the Developer Portal

To learn how to turn content on or off in the Developer Portal, use the following topic links:

- [Deleting a site comment](#)  
You can delete site comments in the Developer Portal.
- [Deleting blocks](#)  
You can delete blocks in the Developer Portal.
- [Disabling blogs](#)  
You can disable blogs on your Developer Portal.
- [Deleting forums from the front page](#)  
You can delete the help information on forums from the front page of your Developer Portal.
- [Turning comments for specific content types on or off in the Developer Portal](#)  
You can turn the comments for specific content types on or off in the Developer Portal.
- [Turning comments off for an individual item](#)  
You can turn off comments for individual items within a content type.
- [Turning off ratings for specific content types in the Developer Portal](#)  
You can turn off ratings for specific content types in the Developer Portal by hiding the ratings field.
- [Turning the ability to tag specific content types off in the Developer Portal](#)  
You can turn the ability to tag specific content types on or off in the Developer Portal.

---

## Deleting a site comment

You can delete site comments in the Developer Portal.

### Before you begin

---

You must have administrator or content author access to complete this task.

### Procedure

---

To delete a site comment, complete the following steps

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. Click the Comments tab.
4. From the UPDATE OPTIONS list, select Delete the selected comments.
5. Select the comments that you want to delete, then click Update.  
The confirmation screen displays.
6. Click Delete comments.

You have deleted the selected comments.

---

## Deleting blocks

You can delete blocks in the Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

### Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Pages.
4. In the OPERATIONS column, find the welcome page row and select Edit from the drop-down list.
5. Expand Panels, then click Content.
6. In the OPERATIONS column, find the block that you want to delete and select Delete for the drop-down list.
7. Click Delete to confirm, then click Update and save.

---

## Disabling blogs

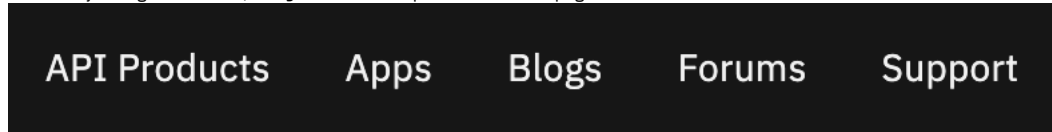
You can disable blogs on your Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

Before any changes are made, **Blogs** show as an option on the front page like this:



### Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure > Menus from the administrator dashboard.
3. In the OPERATIONS column, click Edit menu for **Main navigation**.
4. Clear Enabled for the **Blogs Menu link**, then click Save.

**Title \***

Machine name: main

**Administrative summary****Menu language**

MENU LINK	ENABLED	OPERATIONS
<a href="#">Home</a>	<input checked="" type="checkbox"/>	<a href="#">Edit</a>
<a href="#">API Products</a>	<input checked="" type="checkbox"/>	<a href="#">Edit</a>
<a href="#">Apps</a>	<input checked="" type="checkbox"/>	<a href="#">Edit</a>
<a href="#">Blogs</a>	<input type="checkbox"/>	<a href="#">Edit</a>
<a href="#">Forums</a>	<input checked="" type="checkbox"/>	<a href="#">Edit</a>
<a href="#">Support</a>	<input checked="" type="checkbox"/>	<a href="#">Edit</a>

[Save](#)

## Results

You removed the **Blogs** option from the front page. The options on the front page of your Developer Portal now look like this:



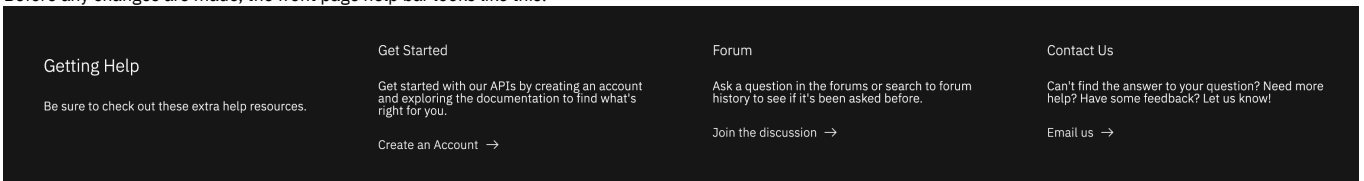
## Deleting forums from the front page

You can delete the help information on forums from the front page of your Developer Portal.

### Before you begin

You must have administrator access to complete this task.

Before any changes are made, the front page help bar looks like this:



## Procedure

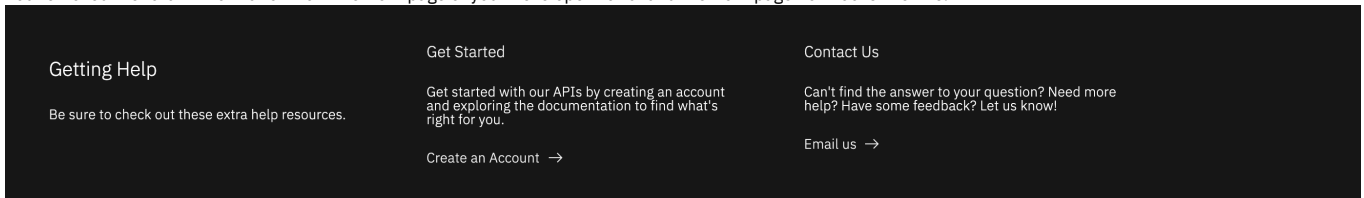
1. If the administrator dashboard is not displayed, click **Manage** to display it.
2. Click **Structure** in the administrator dashboard.
3. Click **Block layout**, Custom block library.
4. In Block description, enter `Get help`, then click **Apply**.

BLOCK DESCRIPTION	BLOCK TYPE	UPDATED	OPERATIONS
Get Help [en]	Basic block	05/07/2020 - 19:52	Edit

- In the OPERATIONS column, click Edit for **Get Help**.
- In the **Body** section, delete the forum information, then click Save.

## Results

You removed the forum information from the front page of your Developer Portal and the front page now looks like this:



## Turning comments for specific content types on or off in the Developer Portal

You can turn the comments for specific content types on or off in the Developer Portal.

### Before you begin

You must have administrator access to complete this task.

### Procedure

To turn comments for specific content types on or off in the Developer Portal, complete the following steps:

- If the administrator dashboard is not displayed, click Manage to display it.
- Click Structure in the administrator dashboard.
- Click Content types.
- In the OPERATIONS column, click Manage fields for the required content type.
- In the OPERATIONS column, click Edit for the Comments field.
 

Note:

  - If there is no Comments field, you can enable commenting for the selected content type by adding one as follows:
    - Click Add field.
    - From the Add a new field list, select Comments.
    - Enter a field label, then click Save and continue.
    - Select the required comment type, then click Save field settings.
  - You can completely disable commenting for the selected content type, and permanently remove all current comments, by selecting Delete.
- Select the required option in the DEFAULT VALUE section.
  - To allow commenting, select Open.
  - To keep existing comments but disallow further commenting, select Closed.
  - To hide all comments, select Hidden.
- When done, click Save settings.

## Turning comments off for an individual item

You can turn off comments for individual items within a content type.

### Before you begin

Users with edit permission for the selected content type can perform this task.

### Procedure

To turn off comments for a single item, complete the following steps:

- If the administrator dashboard is not displayed, click Manage to display it.
- Click Content in the administrator dashboard.
- In the OPERATIONS column, click Edit for the required content item.
- Click Comment settings in the side pane.
- Select Closed to turn comments off for the selected item.

6. Click Save.

---

## Turning off ratings for specific content types in the Developer Portal

You can turn off ratings for specific content types in the Developer Portal by hiding the ratings field.

---

### Before you begin

You must have administrator access to complete this task.

---

### Procedure

To turn off ratings for specific content types in the Developer Portal, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Content types.
4. In the OPERATIONS column for the required content type, select Manage fields from the drop-down list.
5. Click on the Manage form display tab. In the FIELD list, select Rating and drag and drop it into the Disabled section, then click Save.
6. Click on the Manage display tab. Content items are displayed by using different view modes, such as Default, Card, and Teaser, and these modes are shown at the beginning of the Manage display page. For each view mode that is applicable to the content type you're editing, click the mode, and in the FIELD list select Rating, and drag and drop it into the Disabled section.
7. Click Save.

---

## Turning the ability to tag specific content types off in the Developer Portal

You can turn the ability to tag specific content types on or off in the Developer Portal.

---

### Before you begin

You must have administrator access to complete this task.

---

### Procedure

To turn the ability to tag specific content types on or off in the Developer Portal, complete the following steps:

1. Click Structure in the administrator dashboard.
2. Click Content types.
3. Click Manage fields next to the required content type.
4. From the Tags field, click Edit.
5. Click Delete, then confirm the Delete as the action cannot be undone.  
You have turned the ability to tag specific content types off.

---

## Structure

Control the layout of the Developer Portal, such as configuring the front page, adding and changing blocks, and displaying Products and APIs in categories.

To learn how to control the layout of the content on your Developer Portal site, use the following topic links.

- [Configuring the front page](#)  
You can configure the Developer Portal front page.
- [Adding a menu](#)  
You can add a menu to the Developer Portal, and define the items that are included in the menu.
- [Adding and changing the blocks displayed on Developer Portal pages](#)  
You can add new blocks, and modify the existing blocks that are displayed on Developer Portal pages.
- [Changing the front page banner block](#)  
You can change the front page banner content and image that is displayed on the home page when a user first logs in to the Developer Portal.
- [Changing the front page Featured Content block](#)  
You can display featured Products or APIs on the front page of your Developer Portal by configuring the Featured Content block.
- [Changing the items in the main menu](#)  
You can change the items in the main menu that is displayed on all pages in the Developer Portal.
- [Changing the ratings display](#)  
You can change the appearance of the content ratings in the Developer Portal.
- [Displaying APIs and Products in categories](#)  
You can create taxonomies in the Developer Portal, which dictate the categorization of APIs and Products, and how they are displayed. The taxonomies that you define can be used by API developers to categorize APIs and Products in YAML files.
- [Implementing an image carousel](#)  
You can customize your Developer Portal home page to display images in a carousel. The image carousel provides your Developer Portal home page with a continuous slide show that is customized to your specifications.

- [Making your application's image public](#)  
You might want to display your application's image in an oauth authentication page to reassure customers that they are authenticating the correct application in the Developer Portal. That authentication uses a URL to the image that is stored on the Developer Portal. However, by default, those images are private unless you are logged in, which doesn't work if you want to display it in an oauth page. The solution is for the admin to change the `application_image` field to use public instead of private.
- [Providing navigation by tag hierarchy](#)  
You can provide the ability for users in the Developer Portal to navigate by using tag hierarchy.
- [Using the Views module in the Developer Portal](#)  
By using the Views module, you can fetch content from your Developer Portal site, and present it to users in different formats such as lists, graphs, and tables.

---

## Configuring the front page

You can configure the Developer Portal front page.

---

### Before you begin

You must have administrator access to complete this task.

---

### About this task

Create a customized welcome page for your users. Note that blocks that are placed on the front page of the Developer Portal are visible to all users, regardless of whether the blocks have access restrictions placed on them. The visibility of blocks to all users also extends to the Featured Content block.

---

### Procedure

To configure the front page, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure, then click Pages.
3. Alongside the welcome entry, click Edit.
4. Use the options provided to configure the front page.  
Here you can edit the page information, access, and configuration of the welcome page.
5. Click Update and save to save your changes.

---

### What to do next

You can create a custom page. For more information, see [Creating custom pages](#).

- [Disabling blogs](#)  
You can disable blogs on your Developer Portal.
- [Deleting forums from the front page](#)  
You can delete the help information on forums from the front page of your Developer Portal.

---

### Related tasks

- [Changing the front page banner block](#)
- [Changing the front page Featured Content block](#)

---

## Disabling blogs

You can disable blogs on your Developer Portal.

---

### Before you begin

You must have administrator access to complete this task.

Before any changes are made, **Blogs** show as an option on the front page like this:



---

### Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure > Menus from the administrator dashboard.
3. In the OPERATIONS column, click Edit menu for **Main navigation**.
4. Clear Enabled for the **Blogs Menu link**, then click Save.



**Title \***

Main navigation

Machine name: main

**Administrative summary**

Site section links

**Menu language**

English

MENU LINK	ENABLED	OPERATIONS
Home	<input checked="" type="checkbox"/>	Edit
API Products	<input checked="" type="checkbox"/>	Edit
Apps	<input checked="" type="checkbox"/>	Edit
Blogs	<input type="checkbox"/>	Edit
Forums	<input checked="" type="checkbox"/>	Edit
Support	<input checked="" type="checkbox"/>	Edit

Save

## Results

You removed the **Blogs** option from the front page. The options on the front page of your Developer Portal now look like this:



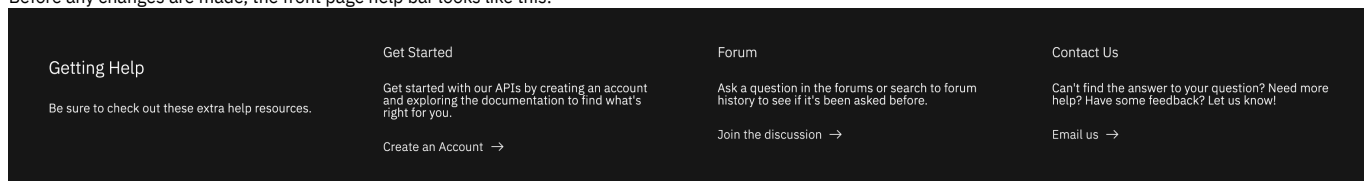
## Deleting forums from the front page

You can delete the help information on forums from the front page of your Developer Portal.

### Before you begin

You must have administrator access to complete this task.

Before any changes are made, the front page help bar looks like this:



## Procedure

1. If the administrator dashboard is not displayed, click **Manage** to display it.
2. Click **Structure** in the administrator dashboard.
3. Click **Block layout**, Custom block library.
4. In **Block description**, enter `Get help`, then click **Apply**.

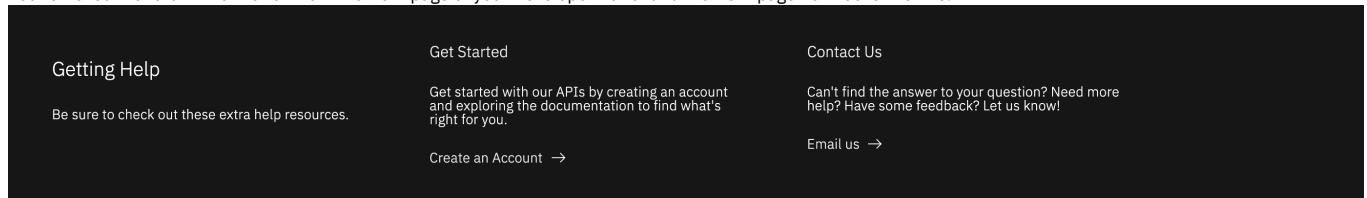
Block description:  Block type:

BLOCK DESCRIPTION	BLOCK TYPE	UPDATED	OPERATIONS
<a href="#">Get Help [en]</a>	Basic block	05/07/2020 - 19:52	<input type="button" value="Edit"/>

- In the OPERATIONS column, click Edit for **Get Help**.
- In the **Body** section, delete the forum information, then click Save.

## Results

You removed the forum information from the front page of your Developer Portal and the front page now looks like this:



## Adding a menu

You can add a menu to the Developer Portal, and define the items that are included in the menu.

### Before you begin

You must have administrator access to complete this task.

### Procedure

To add a menu, complete the following steps:

- If the administrator dashboard is not displayed, click Manage to display it.
- Click Structure > Menus.
- Click Add menu, provide a title, and an optional summary and menu language, and click Save.
- To add items to your menu, click Add link, and configure the menu link details that you require.
- Click Save to save your changes.
  - A new block for your menu is created automatically, and your menu is now in the **Menus** list.

### What to do next

You can use block configuration options to control which pages your menu is displayed on, and where it is positioned. You can use your menu with, or instead of, other menus available within the site. For more information, see [Changing the blocks displayed on Developer Portal pages](#).

You can also update an existing menu, or change your menu. For more information, see [Changing the items in the main menu](#).

## Adding and changing the blocks displayed on Developer Portal pages

You can add new blocks, and modify the existing blocks that are displayed on Developer Portal pages.

### Before you begin

You must have administrator access to complete this task.

### About this task


Blocks are boxes of content that are rendered into an area, or region, of a page. The content can be custom HTML, Github Markdown, or plain text, and you can specify the appearance, size, position, and also configure which pages a block appears on by editing its visibility settings. You can also modify an existing block to change its appearance, size, position, and visibility.

- [Add a new block](#)
- [Edit the content of an existing block](#)
- [Modify the layout or visibility of an existing block](#)

### Procedure

To add a new basic block, complete the following steps.

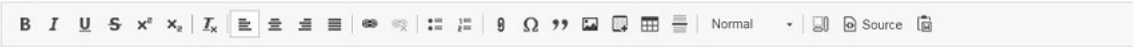
1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure, then click Block layout.  
All of the blocks that are in the system are displayed by theme. From the Block layout page you can assign blocks to a region, and control the order of the blocks within the region. Blocks are positioned on a per-theme basis, so if you enable more than one theme on your site, you can place blocks differently for each theme.
3. Click Place block > Add custom block.  
The Add custom block page is displayed.
4. Complete the Block description field.
5. Enter the content that you require for your block into the Body section.  
You can set the text format to Full HTML, Basic HTML, Restricted HTML, or Github Markdown. You can use the edit icons in the Body section to configure your

content. Click the Source icon  to enter code source, such as HTML.

**Block description \***

A brief description of your block.

**Body**



body p

Text format **Full HTML**

[About text formats ?](#)

6. Click Save.  
The new custom block is saved and the Configure block page is displayed.
7. Complete the block Title field.
8. In the Visibility section, specify how you want the block to be displayed.  
You can specify the content types, configure the language settings, restrict the block to show only on pages that display specific content types, set which pages the block is displayed on, and restrict which user roles the block is visible to.

**Visibility**

<b>Content type</b>	<b>When the user has the following roles</b> <input type="checkbox"/> Anonymous user <input type="checkbox"/> Superuser <input type="checkbox"/> Authenticated user <input type="checkbox"/> Forum Moderator <input type="checkbox"/> Content Author <input type="checkbox"/> Administrator
<b>Language</b> Not restricted	
<b>Content types</b> Not restricted	
<b>Pages</b> Not restricted	
<b>Roles</b> Not restricted	

9. In the Region section, select the region where the block should be displayed.
10. Click Save block.  
The new custom block is added to your Developer Portal site.

To edit the content of an existing block, from the Block layout page complete the following steps.

11. Click the Custom block library tab (or select Structure > Block layout > Custom block library from the administrator dashboard).
12. Click Edit alongside the required block.
13. Change the block description and body content as required.
14. Click Save to save your changes.

To modify the layout or visibility settings of an existing block, from the Block layout page complete the following steps.

15. To change the region in which a block is positioned, select the required Region from the drop-down list next to the required block.
16. To change the vertical sort-order of a block within a region, drag the block to the required position.
17. To edit the visibility settings of an existing block, click Configure from the Operations drop-down menu next to the required block.
18. To enable or disable an existing block, select Enable or Disable from the Operations drop-down menu next to the required block.
19. Click Save blocks to save your changes.

---

## Changing the front page banner block

You can change the front page banner content and image that is displayed on the home page when a user first logs in to the Developer Portal.

### Before you begin

---

You must have administrator or content author access to complete this task.

## About this task


---

You can change the front page banner by editing the Welcome Banner block in the custom block library.

## Procedure

---

To change the front page banner block, complete the following steps.

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure > Block layout.
3. Select the Custom block library tab.
4. Click edit against the Welcome Banner block entry.  
The Edit Basic block Welcome Banner page is displayed.
5. Edit the Body section with the content and image that you require.  
You can set the text format to Full HTML, Basic HTML, Restricted HTML, or Github Markdown. You can use the edit icons in the Body section to configure your content. Click the Source icon  to view and edit the code source, such as HTML.
6. Configure the language and translation settings as required.
7. Click Save to save your changes.

## Results

---

You successfully updated the front page banner block.

## Related tasks

---

- [Configuring the front page](#)
- [Adding and changing the blocks displayed on Developer Portal pages](#)

---

## Changing the front page Featured Content block

You can display featured Products or APIs on the front page of your Developer Portal by configuring the Featured Content block.

## Before you begin

---

You must have administrator access to complete this task.

## About this task

---

You can configure the Featured Content block on the front page of your Developer Portal. By default this block displays a list of the most recently created Products, but you can configure the block to display Products or APIs, and select how the featured content is chosen.

## Procedure

---

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Structure > Pages.
4. Click Edit against the welcome page.  
The edit menu for the Welcome page is displayed.
5. Select Panels > Content from the navigation column, and then click Edit against the Featured Content block.  
The Edit block page for the Featured Content block is displayed.

Configure the node selection settings for this Featured Content block. For each node the block will display its image, title and summary. If no summary is present then a truncated form of the description field will be used instead.

To modify the content shown for a node either Edit that node in the portal or edit the YAML document for that node in the API Manager.

Note that the node selection is per user, and so depending on the visibility settings, different users may see different nodes.

**Block description**  
Featured Content block

**Title \***

Display title

**Node type \***

Feature APIs or Products?

**Number of tiles to display \***

How many tiles should be shown?

**Node selection algorithm \***

Select how the featured content should be chosen. For example, based on creation or modification time, name, or random.

**Custom nodes**

Manually specify the nodes to feature. ';' separated. (This field is only used if using 'Custom' node selection.)

**Region \***

6. Configure the Feature Content block as required.

You can choose to display the title or not, select whether to use Products or APIs, for your node type. You can choose how many tiles to display, and select which algorithm is used to select the node. If you start typing in the **Custom nodes** field, you can choose from the options you are given.

7. Click Update block, then Update and save, to save your changes.

## Results

You successfully configured the Featured Content block on the front page of your Developer Portal.

## Changing the items in the main menu

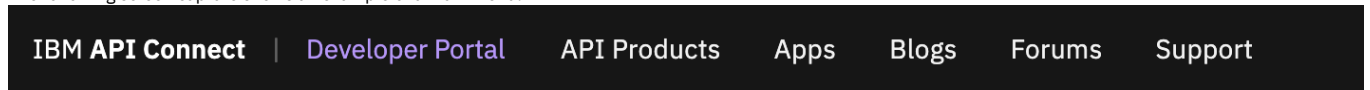
You can change the items in the main menu that is displayed on all pages in the Developer Portal.

## Before you begin

You must have administrator access to complete this task.

## About this task

The following screen capture shows an example of a main menu:



However, you can add new menu items, remove items, and change the item order.

## Procedure

To change the items in the main menu, complete the following steps:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Structure > Menus > Main navigation.

[+ Add link](#)

**Title \***  
 Machine name: main

**Administrative summary**

**Menu language**

MENU LINK	ENABLED	OPERATIONS
<a href="#">+ Home</a>	<input checked="" type="checkbox"/>	<a href="#">Edit</a> ▾
<a href="#">+ API Products</a>	<input checked="" type="checkbox"/>	<a href="#">Edit</a> ▾
<a href="#">+ Apps</a>	<input checked="" type="checkbox"/>	<a href="#">Edit</a> ▾
<a href="#">+ Blogs</a>	<input checked="" type="checkbox"/>	<a href="#">Edit</a>
<a href="#">+ Forums</a>	<input checked="" type="checkbox"/>	<a href="#">Edit</a> ▾
<a href="#">+ Support</a>	<input checked="" type="checkbox"/>	<a href="#">Edit</a>

[Save](#)

4. Use the options provided to change the menu items.
5. Click Save to save your changes.

## Changing the ratings display

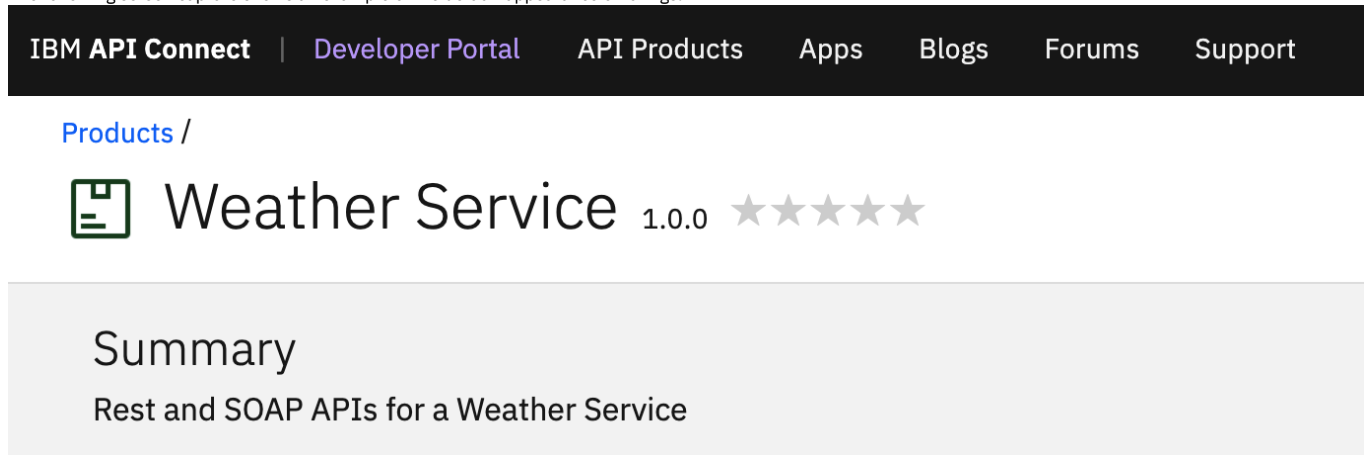
You can change the appearance of the content ratings in the Developer Portal.

### Before you begin

You must have administrator access to complete this task.

### About this task

The following screen capture shows an example of the default appearance of ratings:



You can change this display if you want to.

### Procedure

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Structure > Content types.
4. Click the Manage fields drop-down against the content type you want to change, for example **Product**.
5. Click Edit on the **Rating** row. Then, select the Field settings tab.

These settings apply to the *Rating* field everywhere it is used.

**Vote type \***

**Vote plugin \***

**Allowed number of values**

[Save field settings](#)

6. Change the style settings as required and click Save field settings.  
Here is an example of an alternative Ratings style:

IBM API Connect | [Developer Portal](#) | [API Products](#) | [Apps](#) | [Blogs](#) | [Forums](#) | [Support](#)

[Products](#) /

**Weather Service** 1.0.0 good

**Summary**  
Rest and SOAP APIs for a Weather Service

## Results

You successfully changed the ratings display for a content type in your Developer Portal.

Note: There are different view modes, and the settings would need to be changed for each view mode for each content type.

## Displaying APIs and Products in categories

You can create taxonomies in the Developer Portal, which dictate the categorization of APIs and Products, and how they are displayed. The taxonomies that you define can be used by API developers to categorize APIs and Products in YAML files.

You might want to create taxonomies to control which set of APIs and products can and cannot be used in the Developer Portal.

By default, all taxonomies must be created in the Developer Portal. For more information, see [Managing tags in the Developer Portal](#).

After you create your taxonomies for your APIs and Products, API developers can assign content to those taxonomies manually in the Developer Portal, or by using categories in YAML files.

However, if you create categories for your APIs and Products in the API Designer or API Manager UI, you can have your APIs and Products that are displayed in those categories in the Developer Portal. For more information about how to create categories for your APIs and Products in the API Designer or API Manager UI, see [Organizing your Products into categories](#). For information about how to have your APIs and Products that are displayed in those categories in the Developer Portal, see [Enabling dynamic category creation](#).

You can add more Developer Portal taxonomies to the categories that are defined in the API Designer or API Manager UI.

You can also provide the ability for users in the Developer Portal to navigate by using tag hierarchy. For more information, see [Providing navigation by tag hierarchy](#).

- [Enabling dynamic category creation](#)

You can display APIs and Products in pre-defined categories in the Developer Portal. You can define the categories that the APIs and Products are displayed in, in the API Designer or API Manager UI.

## Enabling dynamic category creation

You can display APIs and Products in pre-defined categories in the Developer Portal. You can define the categories that the APIs and Products are displayed in, in the API Designer or API Manager UI.

## Before you begin

---

You must have administrator access to complete this task.

Your APIs and Products must be categorized in the API Designer or API Manager UI, and they must be published. For more information, see [Organizing your APIs into categories](#) and [Organizing your Products into categories](#).

## About this task

---

By organizing your APIs and Products into categories, you can provide a hierarchical display for your APIs and Products in the Developer Portal.

Note: Organizing APIs and Products into a hierarchical view in the API Manager UI is different from tagging in the Developer Portal.

The categories that are defined in the API Designer or API Manager UI can be overridden by creating taxonomies in the Developer Portal. For more information, see [Displaying APIs and Products in categories](#).

## Procedure

---

1. To enable the Developer Portal to display the categories for your APIs and Products:
  - a. On the administrator dashboard, click Configuration > System > IBM API Developer Portal.
  - b. In the Categories section, select the Create taxonomies for categories if they do not already exist check box.
  - c. Click Save configuration.
2. To enable the Developer Portal to display the change to the categories you made in step 1, you must republish your APIs and products.

## Results

---

You enabled the Developer Portal to display the categories that you defined for your APIs and Products. You can see the categories that you defined, including the number of APIs and Products that are in each category, by selecting your APIs and Products in the Developer Portal.

## Implementing an image carousel

---

You can customize your Developer Portal home page to display images in a carousel. The image carousel provides your Developer Portal home page with a continuous slide show that is customized to your specifications.

## Before you begin

---

You must have administrator access to complete this task.

## About this task

---

By implementing an image carousel, you can replace the default welcome banner or any previously set images.

## Procedure

---

1. Create a content type of Image for your carousel.
  - a. Click Structure > Content types > Add content type.
  - b. Enter the name of your content type in the Name field.  
For example, *Slide show picture*.
  - c. Optional: Specify the description of your content type in the Description field.  
For example, *A picture to display in the slideshow*.
  - d. Click Save and manage fields for your newly created content type.
  - e. Click Add field.
  - f. Select Image for Add a new field, and enter a label for your field.  
For example, *slide image*.
  - g. Click Save and continue.  
The Field settings page for your new field is displayed.
  - h. In the DEFAULT IMAGE section, click Choose file to assign the image that you want as your default image for your slide show if no other images are found.
  - i. Optional: Complete the Alternative text and Title fields.
  - j. Click Save field settings.
  - k. In the Manage fields tab for your newly created content type of *Slide show picture*, delete the Body field by clicking Delete on the OPERATIONS drop-down menu. Click Delete again to confirm.
2. Upload the images that you want to appear in your carousel.
  - a. Click Content > Add content, then click the content type that you added in step 1.
  - b. Enter a title for your image in the Title field.  
For example, if you had an image of a lighthouse, you can label the content as *Lighthouse*.
  - c. Click Choose file under Slide image and select an image for your slide show. Complete the Alternative text field with some text to use when the image cannot be loaded.
  - d. Click Save.
  - e. Repeat sub-steps 2.a to 2.d until you are satisfied with the content of the slides for your carousel.
3. Create a view for your carousel.
  - a. Click Structure > Views > Add new view.
  - b. Complete View name.



- For example, Slide show.
- c. Select the Create a block check box in the BLOCK SETTINGS section.
  - d. Select Slick Carousel for the display format.
  - e. Enter the number of slides you have for your carousel in the Items per block field, then click Save and edit.  
The Edit page for your slide show view is displayed.
  - f. Click the title of the view and enter <none> into the text field. Click Apply.
  - g. In the FIELDS section click Add, select the check box for slide image, and click Add and configure fields.
  - h. Ensure the check box for Create a label is not selected, and click Apply.
  - i. In the FIELDS section, click Content: Title, Remove.
  - j. In the FORMAT section, click Settings for Slick Carousel, and then set Skin Main to Default.
  - k. Scroll down to the CAPTION FIELDS section, and select the Content: slide image and Override main optionset check boxes.
  - l. In the OVERRIDABLE OPTIONS section, select the Autoplay, Dots, and Draggable check boxes.
  - m. Click Apply.
  - n. In the FILTER CRITERIA section, click Content: Published, Remove.
  - o. Click Save to save the Slide show view.
4. Configure your home page to host the carousel.
    - a. Click Structure, then click Pages.
    - b. Click edit for the welcome page, then click content in the Panels section.
    - c. To remove the current welcome banner, click the down-arrow under the OPERATIONS column for the Welcome Banner, and select Delete. Click Delete again to confirm.
    - d. To add your carousel, click Add new block and select the slide show view you created in step 3. Deselect Display title and click Add block.
    - e. Click Update and save.

---

## Results

Your Developer Portal front page now has a slide show carousel.

---

## Making your application's image public

You might want to display your application's image in an oauth authentication page to reassure customers that they are authenticating the correct application in the Developer Portal. That authentication uses a URL to the image that is stored on the Developer Portal. However, by default, those images are private unless you are logged in, which doesn't work if you want to display it in an oauth page. The solution is for the admin to change the `application_image` field to use public instead of private.

---

## Before you begin

You must have administrator access to complete this task.

---

## About this task

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Structure > Content types > Application > Manage fields
4. Navigate to the Image label, then in the Edit drop down, select **storage settings**.
5. Click **Public files** for **Upload destination**.
6. Click **Save field settings**.

---

## Providing navigation by tag hierarchy

You can provide the ability for users in the Developer Portal to navigate by using tag hierarchy.

---

## Before you begin

You must have administrator access to complete this task.

---

## About this task

The Categories block provides the ability to navigate by tag hierarchy. The Categories block is disabled by default, so you need to enable the block and then configure it to your requirements.

---

## Procedure

To provide navigation by tag hierarchy, complete the following steps:

1. Click Extend, in the filter box enter Hierarchical Taxonomy Menu.
2. Select **Hierarchical Taxonomy Menu**, then click Enable.
3. Click Structure > Block layout.  
All of the blocks in the connect\_theme are displayed.
4. Scroll to the region that you want to add the block to, for example Collapse left, click Place block.
5. Find the **Hierarchical Taxonomy Menu** block in the list and click Place block.
6. Complete any configuration tasks that you require for the **Collapse left** block, including selecting the correct taxonomy to use.
7. Click Save block to save your changes.

You can add new blocks, and modify the existing blocks that are displayed on Developer Portal pages. For more information, see [Adding and changing the blocks displayed on Developer Portal pages](#).

## Related tasks

---

- [Editing tags for a specific item](#)
- [Managing tags in the Developer Portal](#)

---

## Using the Views module in the Developer Portal

By using the Views module, you can fetch content from your Developer Portal site, and present it to users in different formats such as lists, graphs, and tables.

The Views module is a powerful SQL query builder that can access almost all the information in your Developer Portal site database and display it in any format.

There are many uses cases for views, but the following list shows some of the common ones:

- You want to display some of the content differently, such as Products, APIs, and applications.
- You want to display a block with the five most recent posts of some kind.
- You want to change the way that articles are displayed.
- You want to provide an unread forum posts list.
- You want to provide a monthly archive of posts.

For information about how to create a view in the Developer Portal, see the following topic. You can also follow a tutorial about creating a custom sort order view; see [Tutorial: Using a custom weighting sort order on the product list page](#).

- [Creating views in the Developer Portal](#)  
You can create new views in the Developer Portal, such as content lists of Products, APIs, and applications, by using the Views module.
- [Configuring the search results view in the Developer Portal](#)  
You can configure the way that the search results are displayed in the Developer Portal.
- [Configuring the default number of items in a view list in the Developer Portal](#)  
You can configure the default number of items that are displayed in a view list in the Developer Portal.

---

## Creating views in the Developer Portal

You can create new views in the Developer Portal, such as content lists of Products, APIs, and applications, by using the Views module.

### Before you begin

---

You must have administrator access to complete this task.

### About this task

---

Creating views in the Developer Portal enables you to control the presentation of specific content for users. For more information about views, see [Using the Views module in the Developer Portal](#).

### Procedure

---

To create a view, complete the following steps.

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. On the administrator dashboard, click Structure > Views > Add new view.  
The Add view page is displayed.
4. Complete the View name and, optionally, the Description.
5. Complete the VIEW SETTINGS section with details of what you want your view to show, and how you want it to be sorted.  
For example, to create a view of APIs that are sorted by title, select Show Content, of type API, and sorted by Title.
6. Select whether to Create a page, Create a block, or both, for your view.  
A page is a full-screen page in the Developer Portal. Whereas a block would be a site block that you can then place on whatever pages you want to show a smaller view of the content, for example the 10 newest APIs.
7. Complete the options that you require for your view. For example, the display format, the number of items per page or block, whether to include a pager, whether to create a menu link, and whether to include an RSS feed.
8. Click Save and edit.  
Your new view opens in the Edit tab.
9. Refine your view further.  
Here you can further modify the display of your view or add new displays. More options include formatting of the view, what fields to include in a table or grid format, filtering and sorting criteria, add header or footer details, as well as advanced options around contextual filters, relationships, and behavior when there are no results to display. The Edit tab also includes a preview area, so you can check your view before publishing.
10. Click Save to make your changes permanent.  
When saved, your new view is visible in the Structure > Views pane.

## Results

---

You created a new view in the Developer Portal.

## Configuring the search results view in the Developer Portal

---

You can configure the way that the search results are displayed in the Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

### About this task

---

In the Developer Portal, the search results are displayed as a view, and so you can control the presentation of this search content. For more information about views, see [Using the Views module in the Developer Portal](#).

### Procedure

---

To configure the search results view, complete the following steps.

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click **Manage** to display it.
3. On the administrator dashboard, click **Structure > Views**.  
The list of views is displayed.
4. Find the **Search content view** in the list, and click **Edit**.
5. Edit the **Search content view** settings as required.  
Here you can modify the display of the view or add new displays. For example, you can configure the formatting of the view, the filtering and sorting criteria, add header or footer details, as well as more advanced options around contextual filters, relationships, and behavior when there are no results to display. There is also a preview area so you can check the view before publishing.
6. Click **Save** to make your changes permanent.

## Results

---

You updated the view of the search results in the Developer Portal.

### What to do next

---

You can configure the search index and server that are used to generate the search results. For more information, see [Configuring search indexes and servers](#).

### Related information

---

- [Drupal Views module](#)

## Configuring the default number of items in a view list in the Developer Portal

---

You can configure the default number of items that are displayed in a view list in the Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

### About this task

---

In the Developer Portal, you can manage how many items in a list are shown for each page on a view. For more information about views, see [Using the Views module in the Developer Portal](#).

### Procedure

---

To configure the search results view, complete the following steps.

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click **Manage** to display it.
3. On the administrator dashboard, click **Structure > Views**.  
The list of views is displayed.
4. In the list, find the view name that you want to update, for example, **Products**, then click **Edit**.
5. Click **Pager**.
6. Select **Display a specified number of items**, click **Apply**.

7. Set the Items per page as required, click Apply.
8. Click Save to make your changes permanent.

## Results

---

You updated the default number of items in a view of the search results in the Developer Portal.

## What to do next

---

The Views module is a powerful SQL query builder that can access almost all the information in your Developer Portal site database and display it in any format. For more information, see [Using the Views module in the Developer Portal](#).

## Configuration

---

As an administrator, you can control multiple aspects of the Developer Portal configuration, including managing security and general configuration tasks.

To learn how to configure specific aspects of your Developer Portal site, use the following section links.

- [Configuring content moderation](#)  
By configuring content moderation, you can configure a review and approval process for content types in the Developer Portal. You can specify which roles can perform which actions by assigning them the appropriate permissions.
- [General configuration tasks](#)  
You can perform general configuration tasks in the Developer Portal as an administrator.
- [Managing Developer Portal security](#)  
You can manage multiple security elements in the Developer Portal.

## Configuring content moderation

---

By configuring content moderation, you can configure a review and approval process for content types in the Developer Portal. You can specify which roles can perform which actions by assigning them the appropriate permissions.

## Before you begin

---

An example use case for configuring content moderation would be for a professional documentation writer to review an APIs documentation before it is published on the Developer Portal. They might, for example, notify the API Developer of typographical errors, suggest improvements to operation descriptions, or attach an image file. Edits might then be made before the documentation was republished to the API and then published to the Developer Portal.

You must have a Developer Portal enabled, and you must have administrator access to complete this task.

## About this task

---

By using content moderation, you can configure the workflow of your Developer Portal site so that you can moderate content types, such as APIs and Products. However, you cannot moderate Consumer organizations or applications.

## Procedure

---

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Enable the required modules:
  - a. On the administrator dashboard, click Extend.
  - b. In the search bar enter **workflows**, then select the Workflows module and click Enable.
  - c. In the search bar enter **content moderation**, then select the Content Moderation module and click Enable.
4. Navigate to Configuration > Workflow > Workflows
5. Check that **Editorial** is listed as a workflow. If it isn't listed:
  - a. Click + **Add workflow**.

**Label \***

 Machine name: editorial [Edit]

**Workflow type \***

**Save**

- b. Enter **Editorial** as the label.
  - c. Select the workflow type **Content moderation**.
  - d. Click **Save**.
6. Set up the workflow:
    - a. Click **Edit** on the **Editorial** workflow.

b. Click the **Add a new state** link.

**Label \***

 Machine name: editorial

c. Enter the value **Review** in the **State label** field, leave the **Published** and **Default revision** check boxes cleared then click Save.

**State label \***

 Machine name: review [Edit]

**Published**  
When content reaches this state it should be published.

**Default revision**  
When content reaches this state it should be made the default revision; this is implied for published states.

**Save**

d. Drag the **review** state to between the **Draft** and **Published** states then click 'Save'. If the drag handles are not visible, then click the **Hide row weights** link in the **STATES** section to make them appear.

**Label \***

 Machine name: editorial

▼ STATES

⚠ \* You have unsaved changes.

STATE	OPERATIONS
<input type="checkbox"/> Draft	<b>Edit</b>
<input type="checkbox"/> Review*	<b>Edit</b> ▼
<input type="checkbox"/> Published	<b>Edit</b>

[Add a new state](#)

e. Click the **Add a new transition** link.

f. Enter the value **Review** in the **Transition Label** field, tick the **Draft** check box in the **From** section and click **Review** in the **To** section then click Save.

g. Drag the **Review** transition to between the **Create New Draft**

**Draft** and **Publish** transitions, then click Save.

h. Click the **Edit** operation on the **Create New Draft** transition.

i. In the **From** check box list, untick **Published** and tick **Review** then click Save.

j. Click the **Edit** operation on the **Publish** transition.

k. In the **From** check box list, untick **Draft** and tick **Review** then click Save.

l. Click **Select** for the **Content types** item in the **THIS WORKFLOW APPLIES TO** section.

m. Select the **Product** content type then click Save.

7. Set up the permission schema for the workflow:

a. Click **People > Roles > Add a new role**.

b. In the **Role Name** field, enter the value **Editor** then click Save.

c. Drag the **Editor** role to between the **Content Author** and **Administrator** roles then click Save.

- d. Click the Permissions tab and scroll down to the **Content Moderation** permission section.
  - e. For the EDITOR role, select all the permissions in the **Content Moderation** section.
  - f. For the CONTENT AUTHOR role, select the permissions **Editorial workflow: Use Create New Draft transition**, **View any unpublished content**, and **View the latest version** then click Save permissions.
  - g. Ensure that the EDITOR and CONTENT AUTHOR roles have the permission **Use the administration toolbar**.
8. Assign the permissions to users in the portal. If you do not already have some registered users in your portal, create a new consumer organization and invite two new users to it.
- a. Click People.
  - b. Tick the check box next to one of the users in the list, but not the admin user, then from the **Action** drop-down select **Add the Content Author role to the selected user(s)** then click **Apply to selected items**.
  - c. Tick the check box next to another one of the users in the list, but not the admin user, then from the **Action** drop-down select **Add the Editor role to the selected user(s)** then click **Apply to selected items**.

---

## Results

You successfully configured content moderation on your Developer Portal. You can now use the API manager user interface to publish a product to the portal, and transition the product from draft to review state.

- [Editing, reviewing, and publishing moderated content](#)  
If you have published content from API Manager UI to your Developer Portal or created content within the Developer Portal, you can edit and review the content through the Moderated content page before you publish it in the Developer Portal.

---

## Editing, reviewing, and publishing moderated content

If you have published content from API Manager UI to your Developer Portal or created content within the Developer Portal, you can edit and review the content through the Moderated content page before you publish it in the Developer Portal.

---

## Before you begin

You must have administrator access to complete this task.

You must have content that is published to the Developer Portal from the API Manager UI.

Your role must have the necessary permissions that enable you to perform the tasks. For more information on assigning the necessary tasks, see [Configuring content moderation](#).

---

## About this task

In addition to editing content through the content moderation page, you can specify whether the content needs further review, or even publish the content if you have the permission.

---

## Procedure

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. On the administrator dashboard, click Content.
4. Click the Moderated content tab.
5. If you have the necessary permissions to edit your content type, for example, you are a **Content author**:
  - a. Click the Title of the content type that you want to edit, then click the Edit.
  - b. Edit your content to your specification, then click Save.  
You are redirected to the **View** tab.
  - c. When you have finished editing your content type on the **edit** tab, ensure that the **Review** state is selected from the **Change to** drop down, then click Save.
6. If you have the necessary permissions to review and publish your content type, for example, you are a **Reviewer**:
  - a. On the administrator dashboard, click Content.  
The content page is displayed listing all content.
  - b. Click the Moderated content tab.
  - c. Click the Title of the content type that you want to review.
  - d. Optional: If the content that you are reviewing requires editing, click Edit, apply the changes to your content type, then click Save.  
You are redirected to the View tab.

---

## Results

You edited, reviewed, and published your draft content type.

---

## General configuration tasks

You can perform general configuration tasks in the Developer Portal as an administrator.

To learn how to configure specific aspects of your Developer Portal site, use the following topic links.

- [Adding a page load progress indicator](#)  
You can extend your Developer Portal site by adding a page load progress indicator. This option might be useful on servers that have a slow connection to the API Manager, or for customers that are using OIDC and have slow response times from their OIDC provider.
- [Adding custom fields to user records](#)  
You can add custom fields to user records in the Developer Portal.
- [Checking the site status report](#)  
You can check the site status report in the Developer Portal.
- [Clearing the server caches](#)  
You can clear the server caches from within the Developer Portal.
- [Configuring a proxy](#)  
You can configure the proxy settings in your Developer Portal.
- [Configuring Stripe in the Developer Portal](#)  
You can offer free use of your Product Plans to your API consumers, or you can offer Products with paid Plans. To enable API consumers to subscribe to Products with paid Plans, you must first configure the Stripe payment method in your Developer Portal.
- [Configuring cron to run scheduled tasks](#)  
You can configure cron to run scheduled tasks in the Developer Portal.
- [Configuring search indexes and servers](#)  
You can configure search indexes and servers in the Developer Portal by using the Search API module.
- [Configuring the date and time](#)  
You can configure the date and time in the Developer Portal.
- [Configuring the notifications event log settings](#)  
You can configure the settings of the notifications event log for the activity feed on your Developer Portal.
- [Configuring the settings of a Consumer organization](#)  
You can configure the settings of your Consumer organization from within your Developer Portal. The default settings allow users to delete their accounts and organizations, create more consumer organizations, rename their organization, and change the owner of their organization.
- [Configuring the site default timezone](#)  
You can configure the site default timezone in the Developer Portal.
- [Configuring the site error handling](#)  
You can configure the site error handling in the Developer Portal.
- [Configuring which buttons are displayed in the WYSIWYG rich text editor](#)  
You can configure which buttons are displayed in the WYSIWYG rich text editor in the Developer Portal.
- [Configuring which languages are available](#)  
You can configure which languages are available in the Developer Portal.
- [Customizing user account settings](#)  
You can customize the user account options that are displayed to users when they register and use the Developer Portal.
- [Disabling languages](#)  
You can extend your Developer Portal site by managing which languages are available for use on the site.
- [Disabling test tool restrictions](#)  
The default OAuth provider contains security mechanisms that prevent the ability to obtain authorization tokens in the Developer Portal test tool when you use Implicit or Authorization Code grant types. If your OAuth provider allows those actions to function correctly from the test tool, you can disable this option.
- [Enabling a cookie compliance banner](#)  
You can add a cookie compliance banner to your Developer Portal site by configuring the EU Cookie Compliance module.
- [Enabling code languages for code snippets](#)  
You can specify the languages that code snippets for APIs can be displayed in.
- [Enabling code snippets for SOAP APIs](#)  
By default, code snippets are only shown for REST APIs. You can enable code snippets for SOAP APIs.
- [Forcing new users to accept terms and conditions](#)  
You can extend your Developer Portal site by forcing new users to accept the terms and conditions before they are able to create accounts.
- [Hiding the admin registry on the login form](#)  
You can hide the admin registry as an option on the login form. The use case for this option might be a public facing Developer Portal where you want to remove the admin access from customer view.
- [Hiding the certificate in the header for APIs secured with mutual TLS](#)  
You can hide the certificate in the header for APIs secured with mutual TLS. By default, the `x-client-certificate` header is shown in an API when mutual TLS is configured. You can choose to turn off this option in your Developer Portal.
- [Importing and exporting taxonomies](#)  
You can import and export taxonomies in the Developer Portal.
- [Restricting access by IP address](#)  
You can restrict access to a role or a user by IP address in the Developer Portal.
- [Restricting or preventing access from search engines](#)  
You can restrict or prevent access to your Developer Portal from search engines. You might want to control the access if you have a development or test site.
- [Toggling the site in and out of maintenance mode](#)  
You can put the Developer Portal site into maintenance mode for short periods of time. During maintenance mode, only an administrator is able to access the site, and all other users who enter the site URL get a maintenance message set by the administrator.
- [Viewing available updates](#)  
You can view available updates in the Developer Portal.

---

## Adding a page load progress indicator

You can extend your Developer Portal site by adding a page load progress indicator. This option might be useful on servers that have a slow connection to the API Manager, or for customers that are using OIDC and have slow response times from their OIDC provider.

---

### Before you begin

You must have administrator access to complete this task.

## Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. In the administrator dashboard, click Extend.  
The List tab for the Extend page opens, and the list of installed modules is displayed. The list shows all the modules that are installed. The enabled modules are displayed with a selected check box. The disabled modules are displayed without a selected check box.
3. Enter `page_load_progress` into the search filter.  
The **Page Load Progress** content item is displayed in the list of content.
4. By default, the **Page Load Progress** module is enabled. However, if not, select the check box and click Enable.  
You have now enabled the default settings for the **Page Load Progress** module.
5. To change the default settings for the module, navigate to Configuration > User interface > Page Load Progress.
6. Decide which settings you want to change, for example, **Time to wait before locking the screen**, and click Save.  
Note: Changing the settings does not apply to the OIDC buttons. If it is enabled, the OIDC buttons use the page load process indicator, but do not allow any configuration.
7. To enable the permissions, navigate to People > Permissions.
8. Scroll down to **Page Load Progress**, check the boxes for anonymous and authenticated for **Use Page Load Progress**.

Page Load Progress						
Administer Page Load Progress	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Use Page Load Progress	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

9. Click Save permissions.

## Results

You successfully enabled the **Page Load Progress**. Now, the page loading spinner is automatically added to the login and register forms for OIDC buttons.

## What to do next

You can customize the appearance of the modal page and spinner in CSS in a custom theme.

## Adding custom fields to user records

You can add custom fields to user records in the Developer Portal.

## Before you begin

You must have administrator access to complete this task.

## About this task

Any data that is added to a custom field for user records will remain in the Developer Portal database.

## Procedure

1. In the administrator dashboard, click Configuration > People > Account settings.
2. Click the MANAGE FIELDS tab.
3. Click Add field.
4. Specify the type of data that your field can store by selecting the type from the Add a new field list.
5. Enter a label for your field in the Label field.
6. Optional: If you want to edit the machine name that is automatically created based on your label, click Edit and enter the new name.
7. Click Save and continue.
8. Configure any additional values for the new field then click Save field settings.  
You have added a custom field to your user record.

## What to do next

You can configure the contents of your custom field depending on what type of data you specified it could store.

## Checking the site status report

You can check the site status report in the Developer Portal.

## Before you begin

You must have administrator access to complete this task.

## Procedure



To check the site status report, complete the following steps:

1. Click Reports in the administrator dashboard.
2. Click Status report.
3. You can review information relating to the site status report from this view.

---

## Clearing the server caches

You can clear the server caches from within the Developer Portal.

---

### Before you begin

You must have administrator access to complete this task.

---

### Procedure

1. In the administrator dashboard, click Configuration, then in the Development section, click Performance.
2. Click Clear all caches.  
You have cleared the server caches, and a message will be displayed stating: `Caches cleared`.

---

## Configuring a proxy

You can configure the proxy settings in your Developer Portal.

---

### Before you begin

You must have administrator access to complete this task.

---

### About this task

The Developer Portal communicates with the management server by way of the consumer and product APIs. For a Developer Portal that is deployed externally to the management server zone, and does not have access to the consumer and product APIs, a proxy can be used.

---

### Procedure

To configure a proxy to use for communicating with management server APIs, complete the following steps:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Configuration > System > IBM API Connect in the administrator dashboard.
4. To configure a proxy, select **Enable Proxy Support**, and make the required changes.

**PROXY CONFIGURATION (EXPERIMENTAL)**

If a proxy is required to allow communication from the portal server to the management server zone, and does not have access to the consumer and product APIs, a proxy can be used.

Enable Proxy Support

**Proxy type**

CURLPROXY\_HTTP

Select what type of proxy to use, CURLPROXY\_HTTP is the default.

**Proxy URL**

Provide the URL of the proxy e.g. `http://proxyserver.domain.com:8080`.

**Proxy Authentication**

If the proxy requires authentication then provide the 'username:password'.

Save configuration

5. When your changes are complete, click Save configuration.

## Results

---

You successfully configured a proxy on your Developer Portal.

## Configuring Stripe in the Developer Portal

---

You can offer free use of your Product Plans to your API consumers, or you can offer Products with paid Plans. To enable API consumers to subscribe to Products with paid Plans, you must first configure the Stripe payment method in your Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

To use Stripe as your credit card processing vendor, you must have port 443 open to HTTPS communication between the Stripe API and your Developer Portal management and the Management cluster servers. See [Firewall requirements on Kubernetes](#) and [Firewall requirements on VMware](#) for more information about this requirement.

The billing microservice must be enabled on your management system by your system administrator. See [Configuring monetization on VMware](#) and [Configuring monetization on Kubernetes](#) for more information.

Billing integration support must be configured in the API Manager. For more information, see [Monetizing your Products](#).

### About this task

---

API Connect includes a subscription billing feature that allows API providers to define pricing Plans in their API Products, and monetize their API offerings. If a Product contains a pricing Plan, API consumers must enter their payment information into the Developer Portal before they can subscribe to that Plan.

API Connect supports integration with Stripe Subscription Billing, an independent cloud service that manages monetized product Plans, customers, their payment information, and their subscription history, in order to generate monthly invoices and charge customers automatically. With this integration, Stripe serves as both the subscription billing system and the payment processing system.

Support is provided for Stripe in the Developer Portal, but before API consumers can enter their payment information, you need to configure the Stripe billing support in every Developer Portal where you will offer paid Plans. After this support is configured, API consumers can provide their payment information, and then subscribe to the paid Plans. Complete the following instructions to enable the APIC Monetization Stripe Integration module, and then configure your Stripe API credentials.

### Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. In the administrator dashboard, click Extend.  
The List tab for the Extend page opens, and the list of installed modules is displayed. The list shows all the modules that are installed. The enabled modules are displayed with a selected checkbox. The disabled modules are displayed without a selected checkbox.
3. Enter `stripe` into the search filter.  
The `ibm_stripe_payment_method` module called APIC Monetization Stripe Integration is displayed in the list of contents.
4. Select APIC Monetization Stripe Integration and click Enable.  
The default settings for the `ibm_stripe_payment_method` module are now enabled.
5. In the administrator dashboard, click Configuration > System > IBM API Developer Portal Billing.  
The IBM API Developer Portal Billing Settings page is displayed.
6. Change the drop-down option to be the newly enabled APIC Monetization Stripe Integration (`ibm_stripe_payment_method`) as the module to use to create payment methods for the Stripe billing provider. Click Save configuration.
7. In the administrator dashboard, click Configuration > System > IBM APIC Stripe Integration, and enter your Stripe API credentials. These credentials must match the Stripe API credentials for the billing integration resource that is attached to the Catalog for this Developer Portal.  
Refer to your Stripe dashboard to get your publishable key and secret key; see <https://dashboard.stripe.com/apikeys>.  
Note: Each Stripe account comes with two sets of API keys, one for testing, and one for production. Each set of API keys has a distinct namespace for Stripe objects. Test API keys cannot see objects created by production API keys, and vice versa. You cannot switch the API keys of one of your billing integrations with the keys from another account, or swap your test and production keys over, as that would prevent API Connect from resolving the Stripe objects that were created by using the old keys. If you want to use your Stripe test keys, you must create a separate Catalog for testing, and not add your Stripe test keys to production Catalogs.
8. Click Save integration.

## Results

---

You successfully configured the Stripe payment method in the Developer Portal. API consumers can now add a payment method to their consumer organization, so that they can subscribe to Products with paid Plans.

### What to do next

---

After both the Catalog and the Developer Portal are configured, you can create Products with pricing Plans. See [Defining a Product with billing integration](#) for more information.

### Related information

---

- [Monetizing your Products](#)
- [Considerations when changing a Product lifecycle with billing](#)

---

## Configuring cron to run scheduled tasks

You can configure cron to run scheduled tasks in the Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

### Procedure

---

To configure cron to run scheduled tasks, complete the following steps:

1. Click Configuration in the administrator dashboard.
2. Under the SYSTEM heading, click Cron.
3. Use the drop-down list for the module you want to configure and select Edit.
4. From the Run cron every drop-down menu, select the frequency of cron runs for this module.
5. Click Save.

You have configured cron to run scheduled tasks.

---

## Configuring search indexes and servers

You can configure search indexes and servers in the Developer Portal by using the Search API module.

### Before you begin

---

You must have administrator access to complete this task.

### About this task

---

The Search API module provides a generic framework for search capabilities within the Developer Portal, and you can customize the default database server and content index setup that is provided. By default, the search in the Developer Portal runs over the entire OpenAPI and WSDL documentation. It is not possible to restrict the search to specific sections of an Open API or WSDL document. The search can be configured to include any custom fields that were added to the Developer Portal, but it does not include any attached files.

The search results page is a view in the Developer Portal, and so it can also be configured. For more information, see [Configuring the search results view in the Developer Portal](#).

### Procedure

---

To configure the default database server and content index settings, complete the following steps.

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Configuration, Search and metadata, Search API in the administrator dashboard.
4. To configure the server settings, click Edit against the Database Server and make the required changes. For example:
  - Server name; the displayed name for the server.
  - Description; a description of the server.
  - CONFIGURE DATABASE BACKEND; the minimum number of characters a word must consist of to be indexed, and whether to search on parts of a word. However, wildcard searching can make searches on large sites very slow.
  - AUTOCOMPLETE SETTINGS; how suggestions are computed when autocompletion is used.
5. To configure the index settings, click Edit against the Default content index and make the required changes. For example:
  - Index name; the displayed name for the index.
  - Data sources; the data sources of the items to be stored in the index.
  - CONFIGURE THE DATASOURCE; for each data source, set which bundles of content should be indexed, and in which languages.
  - Server; the server that the index should use.
  - Description; a description for the index.
  - INDEX OPTIONS; whether the index is read only, whether to index new or updated items immediately, and setting how many items to index when indexing items during a cron run.
6. When your changes are complete, click Save.  
The View page of the updated default database server or default content index is displayed.

### Results

---

You successfully configured the search indexes and servers in the Developer Portal.

### What to do next

---

You can configure how searches are displayed by editing the Search content view. For more information, see [Configuring the search results view in the Developer Portal](#).

### Related information

---

- [Search API module](#)

---

## Configuring the date and time

You can configure the date and time in the Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

### Procedure

---

To configure the date and time, complete the following steps:

1. If the administrator dashboard is not displayed, click **Manage** to display it.
2. Click **Configuration** in the administrator dashboard.
3. Under the **Regional and language** heading, click **Date and time formats**.
4. Click **Edit** for the format you want to change.
5. Change the settings of the format then click **Save format**.  
You have configured the date and time.

---

## Configuring the notifications event log settings

You can configure the settings of the notifications event log for the activity feed on your Developer Portal.

### About this task

---

The Developer Portal includes an activity feed that shows Consumer organization users the details of events that occur on the organization, such as application creation, subscriptions, and so on. This activity feed is displayed by default on the navigation bar of the organization, and as an additional tab on the organization page and the application page.

The default number of days that events are kept for is 30. You can edit this setting in the IBM APIC Notifications Settings page of your Developer Portal, as described in the following instructions.

### Procedure

---

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click **Manage** to display it.
3. Navigate to **Configuration > System > IBM APIC Notifications Settings**.  
The **IBM APIC Activity Feed Settings** page is displayed.
4. Enter the required number of days to keep events for in the **Retention Days** field.
5. Click **Save configuration**.

---

## Configuring the settings of a Consumer organization

You can configure the settings of your Consumer organization from within your Developer Portal. The default settings allow users to delete their accounts and organizations, create more consumer organizations, rename their organization, and change the owner of their organization.

### Procedure

---

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click **Manage** to display it.
3. Navigate to **Configuration > System > IBM API Connect**.
4. Scroll down to **CONSUMER ORGANIZATIONS**.

#### CONSUMER ORGANIZATIONS

- Allow users to delete their accounts  
If checked then users will be allowed to delete their accounts.
- Allow users to create additional consumer organizations  
If checked then users will be allowed to create additional consumer organizations. Note that self service onboarding must also be enabled in API Manager catalog settings.
- Allow users to rename their organization  
If checked then consumer organization Owner and Administrators will be able to rename their consumer organizations.
- Allow users to change the owner of their organization  
If checked then consumer organization Owner and Administrators will be able to change the owner of their consumer organizations.
- Allow users to delete their organizations  
If checked then Owner or Administrators users will be allowed to delete their consumer organizations.

5. Clear which ever options you want to change, click Save configuration.

---

## Configuring the site default timezone

You can configure the site default timezone in the Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

### Procedure

---

To configure which languages are available, complete the following steps:

1. Click Configuration in the administrator dashboard.
2. Under the REGIONAL AND LANGUAGE heading, click Regional settings.
3. From the Default Time Zone drop-down menu, select the default time zone.
4. Click Save configuration.  
You have configured the default time zone.

---

## Configuring the site error handling

You can configure the site error handling in the Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

### Procedure

---

To configure the site error handling, complete the following steps:

1. Click Configuration in the administrator dashboard.
2. Under the DEVELOPMENT heading, click Logging and errors.
3. You can review and configure all information relating to site error handling from this view.  
Note: On a production system, set Logging and errors to **none**, to avoid any security issues.
4. Click Save configuration.  
You have configured the site error handling.

---

## Configuring which buttons are displayed in the WYSIWYG rich text editor

You can configure which buttons are displayed in the WYSIWYG rich text editor in the Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

### Procedure

---

To configure which buttons are displayed in the WYSIWYG rich text editor, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration > Content authoring > Text formats and editors.
3. Click Configure for the Full HTML profile.
4. In the **TOOLBAR CONFIGURATION** section, edit the buttons that you want to expose.
5. Click Save configuration.  
You have configured the editor buttons that are displayed in the WYSIWYG rich text editor.

---

## Configuring which languages are available

You can configure which languages are available in the Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

### Procedure

---

To configure which languages are available, complete the following steps:

1. Click Configuration in the administrator dashboard.
2. Under the REGIONAL AND LANGUAGE heading, click Languages.
3. Under the DEFAULT heading, select the radio button for the language you want to set as default.
4. To add a new language, click +Add language. Use the drop-down menu to select the language you want to add.
5. Click Save configuration.  
You have configured the languages that are available.
6. You can also provide your own translations, for strings of the website that are not translated, from the User interface translation view, under Configuration > Regional and Language in the administrator dashboard.

### Results

---

You have configured the available languages for the Developer Portal.

Note: Multilingual API and Product documentation can be created by using an **x-ibm-languages** extension directly in the OpenAPI definition. For more information, see [Using x-ibm-languages to create multilingual API and Product documentation](#).

---

## Customizing user account settings

You can customize the user account options that are displayed to users when they register and use the Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

### About this task

---

You can change what fields users of the Developer Portal see when they register to use the site, when they log in, and when they view their user account profile. For instance, you can change the field order display, hide fields, and add custom fields.

As an example, the Consumer organization field is a required entry field when a user creates an account on the Developer Portal. However, you can configure this field to be hidden, in which case the field is automatically completed with a default entry of `firstname lastname` of the new user instead.

### Procedure

---

To customize the user account settings, complete the following steps.

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Configuration > People > Account settings in the administrator dashboard.  
The Account settings page is displayed.
4. From the Settings tab you can edit the following options:
  - a. CONTACT SETTINGS; enable or disable the personal contact form for new users.
  - b. PASSWORD RESET TIMEOUT; set the timeout in seconds for one-time login links. This setting applies only to the admin user of the Developer Portal.
  - c. ANONYMOUS USERS; set the name to be used to indicate anonymous users.
  - d. ADMINISTRATION ROLE; assign the administrator role that is automatically given new permissions whenever a new module is enabled. The default is Superuser.  
Note: The Superuser role has the highest level of permissions in the Developer Portal. This role bypasses all content access control, which means that users that have a Superuser role can see all of the site content. For more information, see [Working with roles in the Developer Portal](#).
  - e. LANGUAGE SETTINGS; enable or disable translation.
  - f. REGISTRATION AND CANCELLATION; enable or disable the password strength indicator, and set the action to take when a user account is canceled.
  - g. Notification email address; set the email address to use as the from address for all account notifications.
  - h. Emails; edit the emails that are sent when an account is blocked or canceled, or when a new password is requested.
  - i. Click Save configuration to save your updates.
5. Click the Manage fields tab to customize the fields that are available for storing user data.

- Here you can edit the settings for current fields, delete fields, and add new fields. Remember to save any updates that you make.
6. Click the Manage form display tab to configure how the user forms are displayed.
    - a. Click the Default form mode to configure how the form fields display when a user profile is being edited.
    - b. Click the Register form mode to configure how the form fields display when a new user creates an account.
 

For each user form mode you can edit the field settings, enable and disable fields, and drag the fields to change their display order. You can also manage the form modes.

For example, to hide the Consumer organization field when a new user creates an account, click the Register form mode, and drag the Consumer organization into the Disabled section.
    - c. Click Save to save your updates.
  7. Click the Manage display tab to configure how fields are displayed on a user profile page.
 

You can edit the field settings, enable and disable fields, and drag the fields to change their display order. You can also manage the display view modes. Remember to save any updates that you make.
  8. Click the Translate account settings tab to configure the language options for the user account settings. Remember to save any updates that you make.

## Results

You successfully customized the user account settings in the Developer Portal.

## Related tasks

- [Configuring CAPTCHA](#)

## Disabling languages

You can extend your Developer Portal site by managing which languages are available for use on the site.

## Before you begin

You must have administrator access to complete this task.

## Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.

To disable all languages apart from English.

2. Navigate to Configuration > Regional and language > Languages > Detection and selection.
3. Select the checkbox for the **Selected language** detection method, and deselect all other methods, then click Save settings.

### Interface text language detection

Order of language detection methods for interface text. If a translation of interface text is available in the detected language, it will be displayed.

DETECTION METHOD	DESCRIPTION	ENABLED	OPERATIONS
+ Account administration pages	Account administration pages language setting.	<input type="checkbox"/>	
+ URL	Language from the URL (Path prefix or domain).	<input type="checkbox"/>	<a href="#">Configure</a>
+ Session	Language from a request/session parameter.	<input type="checkbox"/>	<a href="#">Configure</a>
+ User	Follow the user's language preference.	<input type="checkbox"/>	
+ Browser	Language from the browser's language settings.	<input type="checkbox"/>	<a href="#">Configure</a>
+ Selected language	Language based on a selected language.	<input checked="" type="checkbox"/>	<a href="#">Configure</a>

### Content language detection

Order of language detection methods for content. If a version of content is available in the detected language, it will be displayed.

Customize Content language detection to differ from interface text language detection settings

[Save settings](#)

To disable one or more languages.

4. In the administrator dashboard, click Extend.
 

The List tab for the Extend page opens, and the list of installed modules is displayed. The list shows all the modules that are installed. The enabled modules are displayed with a selected checkbox. The disabled modules are displayed without a selected checkbox.
5. Enter `disable language` into the search filter.
 

The Disable Language content item is displayed in the list of content.
6. Select the checkbox for the **Disable Language** module, and click Enable.
7. Navigate to Configuration > Regional and language > Languages > Detection and selection.
8. Select the checkbox to enable URL, and click Save settings.

### Interface text language detection

Order of language detection methods for interface text. If a translation of interface text is available in the detected language, it will be displayed.

DETECTION METHOD	DESCRIPTION	ENABLED	OPERATIONS
+ Account administration pages	Account administration pages language setting.	<input type="checkbox"/>	
+ URL	Language from the URL (Path prefix or domain).	<input checked="" type="checkbox"/>	<a href="#">Configure</a>

9. Navigate to Configuration > Regional and language > Languages.
10. Click Edit on the language that you want to disable.

11. Select the checkbox to **Disable language**, then click Save language.

**Edit language** ☆

Edit

Home > Administration > Configuration > Regional and language > Languages

**Language code**  
de

**Language name \***  
German

**Direction \***

Left to right

Right to left

Direction that text in this language is presented.

**Disable language**  
This will remove the language from the language switcher and filter out the Simple XML sitemap

**Select language to which we redirect**  
English

This option will redirect to the selected language when a user calls the disabled language

Save language

## Results

You successfully disabled your chosen language.

## Disabling test tool restrictions

The default OAuth provider contains security mechanisms that prevent the ability to obtain authorization tokens in the Developer Portal test tool when you use Implicit or Authorization Code grant types. If your OAuth provider allows those actions to function correctly from the test tool, you can disable this option.

## Before you begin

You must have administrator access to complete this task.

## Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. In the administrator dashboard, click Configuration > System > IBM API Connect.
3. In the **CONFIGURATION** section, deselect the **Optimise OAuth experience in test tool** option

Optimise OAuth experience in test tool

If checked then certain OAuth flows (such as implicit or access code) which cannot be completed from the test tool for technical reasons are optimised to improve usability.

4. Click Save configuration.

## Results

You successfully disabled the **Optimise OAuth experience in test tool** option. You can now use the Developer Portal test tool to test an Implicit or Authorization Code grant type in an OAuth provider API, if your OAuth provider allows those actions to function correctly from the test tool.

## What to do next

You can enable the **Optimise OAuth experience in test tool** option by repeating the task but selecting the option and then clicking Save configuration.

## Enabling a cookie compliance banner

You can add a cookie compliance banner to your Developer Portal site by configuring the EU Cookie Compliance module.

## Before you begin

You must have administrator access to complete this task.

## About this task



The EU Cookie Compliance module provides a customizable pop-up notification that informs visitors to your site that cookies are being used. The notification enables your site visitors to find out more about cookies, and to decide not to browse the site if they disagree with using cookies. You can also provide a link from your cookie compliance banner to an internal or external privacy policy. For more information about configuring an internal privacy policy, see [Customizing the privacy policy statement](#).

The EU Cookie Compliance module is enabled by default. The following instructions show you how to configure the notification.

---

## Procedure

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Configuration > System > EU Cookie Compliance.  
The Settings tab of the EU Cookie Compliance page is displayed.
4. In the SETTINGS section select Enable banner, otherwise the notification does not display.
5. In the PRIVACY POLICY section, edit the Privacy policy link field to provide a link to your privacy policy or other page that explains cookies to your site visitors. The default link is /privacy, which links to the Privacy Policy content that is configured within the Developer Portal. For more information, see [Customizing the privacy policy statement](#).  
External links should start with http:// or https://. For example:  

```
https://example.com/privacy
```
6. Configure the remaining options as required. For example, you can change the wording, and edit the display size and color.
7. Click Save configuration to save your updates.

---

## Results

You successfully enabled a cookie compliance banner on your Developer Portal site. New visitors to your site are now prompted to accept cookies.

---

## Related tasks

- [Customizing the terms of use statement](#)
- [Customizing the privacy policy statement](#)

---

## Enabling code languages for code snippets

You can specify the languages that code snippets for APIs can be displayed in.

---

## Before you begin

You must have administrator access to complete this task.

---

## Procedure

1. On the administrator dashboard, click Configuration > System > IBM API Connect.
2. In the API Code Snippets section, select the check boxes for the languages that you want to make available for the code snippets to be displayed in.
3. Click Save configuration.  
The languages that you have enabled are displayed as options for your code snippets.

---

## Enabling code snippets for SOAP APIs

By default, code snippets are only shown for REST APIs. You can enable code snippets for SOAP APIs.

---

## Before you begin

You must have administrator access to complete this task.

---

## About this task

Code snippets for SOAP APIs use raw HTTP, and do not use any SOAP libraries.

---

## Procedure

1. On the administrator dashboard, click Configuration > System > IBM API Connect.
2. In the API Code Snippets section, select the Display code snippets for SOAP APIs as well as REST APIs check box.
3. Click Save configuration.  
The languages that you have enabled are displayed as options for your code snippets.

---

## Forcing new users to accept terms and conditions

You can extend your Developer Portal site by forcing new users to accept the terms and conditions before they are able to create accounts.

## Before you begin

---

You must have administrator access to complete this task.

Note: From IBM® API Connect Version 10.0.4.0, the **Legal** module is included in the list of core Developer Portal modules. The **Terms of use** module is still available in the Developer Portal, but the **Legal** module provides more configuration options.

## About this task

---

Use the following instructions to enable and configure the **Legal** module for your site terms and conditions.

## Procedure

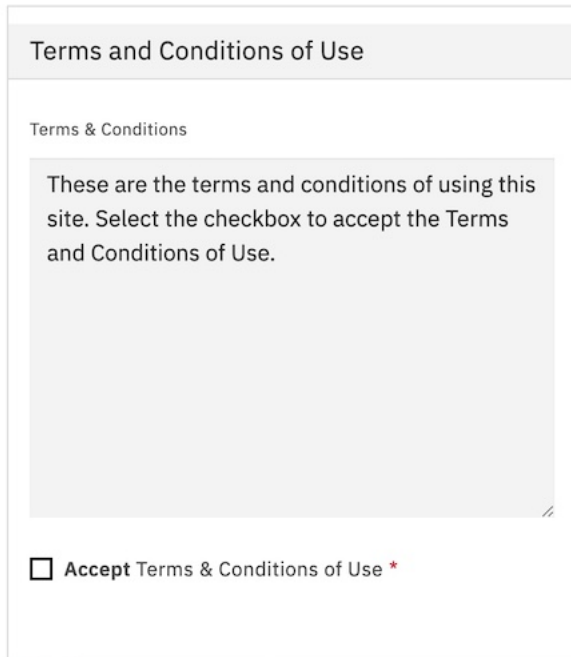
---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. In the administrator dashboard, click Extend.  
The List tab for the Extend page opens, and the list of installed modules is displayed. The list shows all the modules that are installed. The enabled modules are displayed with a selected check box. The disabled modules are displayed without a selected check box.
3. Enter `legal` into the search filter.  
The Legal content item is displayed in the list of content.
4. Select the check box for the **Legal** module, and click Enable.
5. Navigate to Configuration > People > Legal.
6. Enter your Terms & Conditions statement, and select the Text format that you require.
7. Click Save configuration.

## Results

---

You successfully enabled the **Legal** module. Now, when a user tries to create an account on your Developer Portal site, they must agree to the terms and conditions as a part of the registration.



The screenshot shows a registration form titled "Terms and Conditions of Use". Below the title, there is a section labeled "Terms & Conditions" containing a text area with the following text: "These are the terms and conditions of using this site. Select the checkbox to accept the Terms and Conditions of Use." Below the text area, there is a checkbox followed by the text "Accept Terms & Conditions of Use \*".

## What to do next

---

You can edit the terms and conditions page by navigating to Configuration > People > Legal, and making the changes that you require. You can disable the use of terms and conditions during site registration by deleting the terms and conditions content, and then disabling the **Legal** module.

If you prefer to use the **Terms of use** module, you can enable the module, and then navigate to Configuration > People > Terms of use, and select the page that contains your site terms and conditions, for example **Terms of use (1)**. For information about how to customize the statement, see [Customizing the terms of use statement](#). The terms are displayed when you create an account and also from the link in the site footer. You can disable the use of terms and conditions during site registration by disabling the module.

## Related information

---

- [Drupal Legal module](#)

## Hiding the admin registry on the login form

You can hide the admin registry as an option on the login form. The use case for this option might be a public facing Developer Portal where you want to remove the admin access from customer view.

## Before you begin

You must have administrator access to complete this task.

## Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. On the administrator dashboard, click Configuration > System > IBM API Developer Portal.
3. In the Configuration section, select the check box for **Hide the admin registry on the login form**.
4. Click Save configuration.  
The **Hide the admin registry on the login form** configuration is saved.

## What to do next

The admin registry is no longer shown on the login form. However, it is still accessible directly. The user can navigate to `<site_url>/user/login?registry_url=/admin`.

## Hiding the certificate in the header for APIs secured with mutual TLS

You can hide the certificate in the header for APIs secured with mutual TLS. By default, the `x-client-certificate` header is shown in an API when mutual TLS is configured. You can choose to turn off this option in your Developer Portal.

## Before you begin

The parameters for the example API showing the certificate.

Parameters	Header
<b>Accept</b> optional	Permitted values application/json
<b>Content-Type</b> optional	application/json
<b>X-Client-Certificate</b> required	The TLS certificate of your application string

### Example

```
-----BEGIN CERTIFICATE-----xxxEXAMPLExxxxxxxxxEXAMPLExxxxxxxxxxEXA
```

You must have administrator access to complete this task.

## Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. On the administrator dashboard, click Configuration > System > IBM API Developer Portal.
3. In the Configuration section, deselect the checkbox for **Show certificate in header for APIs secured with mutual TLS**.
4. Click Save configuration.  
The **Show certificate in header for APIs secured with mutual TLS** configuration is saved.  
The parameters for the example API now not showing the certificate:

Parameters	Header
<b>Accept</b> optional	Permitted values application/json

# Importing and exporting taxonomies

You can import and export taxonomies in the Developer Portal.

## Before you begin

You must have administrator access to complete this task.

## Procedure

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click **Manage** to display it.
3. In the administrator dashboard, click **Configuration** » **Content authoring** » **Term CSV Export/Import**.
4. Select **Term CSV Import**, or **Term CSV Export**.

For Term CSV Import

5. Enter your CSV input in the format given in the examples, and select the **Taxonomy** from the drop-down list.

The screenshot shows the 'CSV Term Import' interface. At the top, there's a header 'CSV Term Import' with a star icon. Below it are two tabs: 'Term CSV Import' (selected) and 'Term CSV Export'. A breadcrumb trail reads 'Home » Administration » Configuration » Content authoring'. Under the 'Input' label is a large text area for entering CSV data. Below the text area, there's a link 'See CSV Export for an example.' followed by instructions: 'Enter in the form of:' and two examples of CSV formats. The first example is for terms with optional additional fields, and the second is for terms with UUIDs and parent IDs. A note states that additional fields are optional and stringified. There is a checkbox for 'Preserve Vocabularies on existing terms.' which is currently unchecked. Below that is a 'Taxonomy' dropdown menu with 'forums' selected. A 'Next' button is at the bottom of the form.

6. Click **Next**.
7. Click **Import**.

For Term CSV Export

8. Select the **Taxonomy** from the drop-down list, and check any other options that you want.

CSV Term Export ☆

Term CSV Import Term CSV Export

Home » Administration » Configuration » Content authoring

**Taxonomy**

forums ▾

Include Term Ids in export.

Include Term Headers in export.

Include extra fields in export.

Note that fields are stringified using [http\\_build\\_query](#)

Export

9. Click Export.

## Results

---

You successfully imported or exported a taxonomy.

## Restricting access by IP address

---

You can restrict access to a role or a user by IP address in the Developer Portal.

## Before you begin

---

You must have administrator access to complete this task.

## About this task

---

By restricting access to a role by IP address, that role is unavailable to users outside of the IP address ranges that you define. If a role restriction is triggered, the user's session is unaffected, but the restricted role is no longer available to the user. Role restriction affects only the availability of the restricted role to users. Role restrictions are available for all roles, except `anonymous user` and `authenticated user`.

By restricting login access of a user by IP address, the user is unable to log in outside of the IP address ranges that you define. You can also specify global IP address ranges, which apply to all users. IP restrictions are checked on every page load. If a user restriction is triggered by an attempt at logging in being denied, then the user is logged out and sent to the 'error page' that is specified by the site administrator.

Note: IP address ranges must be entered in CIDR notation that is separated with semi-colons and no trailing semi-colon. For more information on CIDR notation, see [CIDR format](#).

## Procedure

---

1. In the Developer Portal, click Configuration, People, Restrict by IP, then click **Restrict log in by IP**.
2. On the **Global restrictions** tab, enter the address of the page to which the user is redirected to if they are not allowed to log in, in the Login denied error page.
3. Select one of the following options:
  - To restrict access to a role by IP address, complete the following steps:
    - Click Restrict role by IP.
    - Decide which role you want to restrict, the roles that you can restrict include Administrator, Content Author, and Forum Moderator.
    - Enter the IP address range (in CIDR notation) that you do not want to restrict log in for in the field for that role, for example, Forum Moderator role IP range.
    - Click Save configuration.

## Restrict role by IP ☆

General Settings

Restrict login by IP

Restrict role by IP

Home » Administration » Configuration » People » Restrict By IP

✓ The configuration options have been saved.

### Forum Moderator role IP range

Enter IP Address Ranges in CIDR Notation separated with semi-colons, with no trailing semi-colon.

For more information on CIDR notation click [here](#).

Leave field blank to disable IP restrictions for Forum Moderator.

### Content Author role IP range

Enter IP Address Ranges in CIDR Notation separated with semi-colons, with no trailing semi-colon.

For more information on CIDR notation click [here](#).

Leave field blank to disable IP restrictions for Content Author.

### Administrator role IP range

Enter IP Address Ranges in CIDR Notation separated with semi-colons, with no trailing semi-colon.

For more information on CIDR notation click [here](#).

Leave field blank to disable IP restrictions for Administrator.

### Superuser role IP range

Enter IP Address Ranges in CIDR Notation separated with semi-colons, with no trailing semi-colon.

For more information on CIDR notation click [here](#).

Leave field blank to disable IP restrictions for Superuser.

Save configuration

- To restrict login access of a user by IP address, complete the following steps:
  - Click Restrict login by IP, then click **User restrictions**.
  - In the **ADD NEW USER ALLOWED IP RANGE** section, enter a user name in the **Username** field.
  - Enter an IP address Range (in CIDR notation), that you do not want to restrict log in for in the **Allowed IP range** field.
  - Click Save configuration. Your new log in restriction can be seen after the **ADD NEW USER ALLOWED IP RANGE** section.

## Restrict login by IP ☆

General Settings

Restrict login by IP

Restrict role by IP

Global Restrictions

User restrictions

Home » Administration » Configuration » People » Restrict By IP » Restrict login by IP

✓ The configuration options have been saved.

### ADD NEW USER ALLOWED IP RANGE

Username

Allowed IP range

Enter IP Address Ranges in CIDR Notation separated with semi-colons, with no trailing semi-colon. E.G. 10.20.30.0/24;192.168.199.1/32;1.0.0.0/8  
For more information on CIDR notation click [here](#).

### apicdemo1 user IP range

Enter IP Address Ranges in CIDR Notation separated with semi-colons, with no trailing semi-colon. E.G. 10.20.30.0/24;192.168.199.1/32;1.0.0.0/8  
For more information on CIDR notation click [here](#).

Leave field blank to disable IP restrictions for apicdemo1.

Save configuration

- To restrict login for all users, complete the following steps:
  - Click Restrict login by IP, then click **Global restrictions** tab.
  - Enter the global IP address ranges (in CIDR notation) in the Restrict global login to allowed IP range field.
  - Click Save configuration.
- 4. To remove an IP restriction, delete the value that is associated with the restriction, then click Save configuration.

## Restricting or preventing access from search engines

You can restrict or prevent access to your Developer Portal from search engines. You might want to control the access if you have a development or test site.

### Before you begin

You must have administrator access to complete this task.

### About this task

You might have a development or test Developer Portal site on the internet, but do not want them to be indexed by Google, or other search engines, making them visible to your customers. Google explicitly advises not to use `robots.txt` as a blocking mechanism. A correct solution is to control access by using Metatag.

### Procedure

1. In the Developer Portal, click Configuration > Search and metadata > Metatag.
2. On the **Global restrictions** tab, click Edit.
3. In the Advanced section, select the meta tags that you want to use to restrict or prevent the access:

## Robots

- index – Allow search engines to index this page (assumed).
- follow – Allow search engines to follow links on this page (assumed).
- noindex – Prevents search engines from indexing this page.
- nofollow – Prevents search engines from following links on this page.
- noarchive – Prevents cached copies of this page from appearing in search results.
- nosnippet – Prevents descriptions from appearing in search results, and prevents page caching.
- noodp – Blocks the [Open Directory Project](#) description from appearing in search results.
- noydir – Prevents Yahoo! from listing this page in the [Yahoo! Directory](#).
- noimageindex – Prevent search engines from indexing images on this page.
- notranslate – Prevent search engines from offering to translate this page in search results.

Provides search engines with specific directions for what to do when this page is indexed.

4. Click Save.

---

## toggling the site in and out of maintenance mode

You can put the Developer Portal site into maintenance mode for short periods of time. During maintenance mode, only an administrator is able to access the site, and all other users who enter the site URL get a maintenance message set by the administrator.

---

### Before you begin

You must have administrator access to complete this task.

Important:

- Maintenance mode is designed for short-term site maintenance; it is not meant for long-term usage. While a site is in maintenance mode, the database is not updated with new content from API Manager. As soon as the maintenance tasks are finished, you must take the site out of maintenance mode.
- Do not log out of the Developer Portal while the site is in maintenance mode, as you will not be able to log back in to the UI. In this case, maintenance mode can be turned off only by using the toolkit CLI; see the following instructions for details.

---

### Procedure

To put the site into maintenance mode, complete the following steps.

1. Click Configuration in the administrator dashboard.
2. Under the DEVELOPMENT heading, click Maintenance mode.
3. Select the Put site into maintenance mode checkbox.
4. Enter a site message, or leave the default site maintenance text.  
This message is what will be seen by users who visit the site while it is in maintenance mode.
5. Click Save configuration.  
Your site is now in maintenance mode.

To take the site out of maintenance mode, complete one of the following steps.

6. If you remained logged in to the site as the admin user during the maintenance period, you can use the Developer Portal UI to disable maintenance mode:
  - a. Click Configuration in the administrator dashboard.
  - b. Under the DEVELOPMENT heading, click Maintenance mode.
  - c. Clear the Put site into maintenance mode checkbox.  
Your site is now available to users, and out of maintenance mode.
7. If you're no longer logged in to the site, you can use the toolkit CLI to disable maintenance mode:
  - a. Log in to the toolkit CLI as a Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
```



```
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`. For full details of the `apic login` command, see [Logging in to a management server](#).

b. Run the following command to disable maintenance mode on your Developer Portal:

```
apic --mode portaladmin drupal-state:set --server management_server --org orgid/name --catalog catalogid/name --state_key system.maintenance_mode --input_format integer --state_value 0
```

Where:

- `management_server` is the endpoint URL of your management server (required).
- `catalogid/name` is the ID or name of the catalog that your site belongs to (required).
- `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
- `--state_key` is the name of the state key (required). Must be set to `system.maintenance_mode`.
- `--input_format` is the input format of the value of the state key. Must be set to `integer` for configuring the maintenance mode.
- `--state_value` is the value to assign to the state key (required). Set to `0` to disable maintenance mode.

For example, to disable maintenance mode for a Developer Portal in the `myOrg` organization, and in the `dev` catalog:

```
apic --mode portaladmin drupal-state:set --server my.management.server.com --org myOrg --catalog dev --state_key system.maintenance_mode --input_format integer --state_value 0
Successfully set state system.maintenance_mode to the value of 0.
```

Your site is now available to users, and out of maintenance mode.

---

## What to do next

You can use the toolkit CLI to enable, and disable, maintenance mode. For more information, see the scenario [How to enable and disable maintenance mode on your Developer Portal](#).

---

## Related tasks

- [Getting started with the Portal CLI commands](#)

---

## Related reference

- [Using the drupal-state commands](#)

---

## Viewing available updates

You can view available updates in the Developer Portal.

---

## Before you begin

You must have administrator access to complete this task.

---

## Procedure

To view available updates, complete the following steps:

1. Click Reports in the administrator dashboard.
2. Click Available updates.
3. You can see what updates are available from this view. However, updates can be applied only by applying the latest available API Connect release; see [Upgrading in a Kubernetes environment](#), or [Upgrading in a VMware environment](#), for more information.

---

## Managing Developer Portal security

You can manage multiple security elements in the Developer Portal.

- [Configuring CAPTCHA](#)  
You can add a CAPTCHA challenge to any of your Developer Portal pages to protect your site from spam.
- [Configuring reCAPTCHA](#)  
You can configure reCAPTCHA on your Developer Portal site to provide more protection from spam and abuse.
- [Configuring session timeout and limit](#)  
You can specify session timeout settings to control when a user is automatically logged out of the Developer Portal. You can also restrict the number of sessions a single user has available to them.
- [Configuring the timeout length for the password reset link](#)  
You can configure the timeout length for the one-time password reset link that is sent from the Developer Portal to the admin user account.
- [Configuring your site password policy](#)  
You can configure your site password policy to define the constraints that are applied to the passwords of your Developer Portal users.

- [Disabling CORS warnings](#)  
You can disable cross-origin resource sharing (CORS) warnings for unenforced APIs in the Developer Portal.
- [Disabling live testing of APIs](#)  
You can disable live testing of APIs in the Developer Portal to restrict exposure of an API.
- [How to enable penetration testing, and information about Developer Portal cookies](#)  
Many security features are available in the Developer Portal, but if you need to do a penetration test of your Developer Portal site, some of these features might block the scan that is required as part of the testing. This topic provides guidance on the security features that you might need to disable to allow penetration testing of your site, as well as information about Developer Portal cookies.
- [How to manage IP security in the Developer Portal](#)  
The Developer Portal offers the ability to perform various IP address security measures, such as adding and removing specific IP addresses from the banned IP address list, automatically banning client IP addresses by using the Drupal Perimeter Defence module, or managing login security by using flood control.
- [Managing banned IP addresses](#)  
You can ban specific IP addresses from accessing your Developer Portal site.
- [Using flood control for login security](#)  
You can configure login security for your Developer Portal by using flood control.
- [Login security](#)  
You can configure the login security for your Developer Portal at an IP and user level. You can also configure login security for Developer Portal contact forms.
- [Using the Security kit](#)  
You can improve the security of your website by configuring various options that are available in the Security Kit module, in the Developer Portal.
- [Using Honeypot for spam protection](#)  
Honeypot protection provides security mechanisms to protect your Developer Portal site from form submission by spam bots. If spam bot activity is detected, form submission is blocked.

---

## Configuring CAPTCHA

You can add a CAPTCHA challenge to any of your Developer Portal pages to protect your site from spam.

---

### Before you begin

You must have administrator access to complete this task.

---

### About this task

The types of CAPTCHA challenges that are available include Image and Math based CAPTCHAs. Both types of CAPTCHA challenges are configurable in the CAPTCHA dialog box. By default, your Developer Portal site is configured with the Image based CAPTCHA challenge enabled.

You can also configure reCAPTCHA on your site. For more information, see [Configuring reCAPTCHA](#).

---

### Procedure

To configure CAPTCHAS, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the PEOPLE section, click CAPTCHA module settings.  
The CAPTCHA settings window opens.
4. Use the CAPTCHA configuration options provided as required. For example, you can complete the following tasks:
  - a. Specify the CAPTCHA challenge type.
  - b. Specify which forms must include a CAPTCHA challenge.
  - c. Add a description to the CAPTCHA.
  - d. Enable CAPTCHA statistics.
5. Click Save configuration to save your changes.

---

## Configuring reCAPTCHA

You can configure reCAPTCHA on your Developer Portal site to provide more protection from spam and abuse.

---

### Before you begin

You must have administrator access to complete this task.

---

### About this task

reCAPTCHA is a CAPTCHA-like system that is designed to establish that a computer user is human. The types of reCAPTCHA that are available include Checkbox, Invisible, and Android.

Note: To use reCAPTCHA the administrator of your Developer Portal is required to sign up to Google's reCAPTCHA API, and your portal server must have internet access.

---

### Procedure

To configure reCAPTCHA, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.

2. Click Configuration > People > CAPTCHA module settings > reCAPTCHA in the administrator dashboard. The reCAPTCHA settings window opens.
3. In the GENERAL SETTINGS section, click register for reCAPTCHA. The Google registration site for reCAPTCHA is displayed.
4. Follow the instructions on Google to register your Developer Portal site, and obtain the Site key and Secret key.
5. Return to the reCAPTCHA settings window in the Developer Portal, and enter the Site key and Secret key that you obtained from Google into the GENERAL SETTINGS section.
6. Optional: Update the WIDGET SETTINGS section as required.
7. Click Save configuration to save your changes.
8. Click the CAPTCHA Settings tab, and in the FORM PROTECTION section change the Default challenge type to reCAPTCHA (from module reCAPTCHA).
9. Click Save configuration.

## Results

---

The reCAPTCHA module is now enabled on your Developer Portal site.

## Related tasks

---

- [Configuring CAPTCHA](#)

## Configuring session timeout and limit

---

You can specify session timeout settings to control when a user is automatically logged out of the Developer Portal. You can also restrict the number of sessions a single user has available to them.

## Before you begin

---

You must have administrator access to complete this task.

## Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. To manage the automated logout settings, complete the following steps:
  - a. Click Configuration in the administrator dashboard.
  - b. In the PEOPLE section, click Automated logout settings.Note: You must check **Enforce auto logout on admin pages** for **autoLogout** to apply when navigating the admin areas of the site.

Enforce auto logout on admin pages

If checked, then users will be automatically logged out when administering the site.

**Save configuration**

- c. Use the session timeout configuration options provided as required. For example, you can complete the following tasks:
    - Specify the length of inactivity time, in seconds, before a user is automatically logged out.
    - Specify session timeout values for individual user roles.  
Note: The default session timeout is 30 minutes.
    - Provide a redirection URL at logout.
    - Provide session expiry message texts.
  - d. Click Save configuration to save your changes.
3. To restrict the number of sessions available to a user, complete the following steps:
    - a. Click Configuration in the administrator dashboard.
    - b. In the PEOPLE section, click Session limit.
    - c. Specify the default maximum number of sessions for a user by entering the number into the Default maximum number of active sessions field.
    - d. Select the required action from the When the session limit is exceeded options.
    - e. From the Logged out message severity list, select the severity of the message that the user receives when they are logged out.
    - f. Optionally, in the ROLE LIMITS section, specify separate session limits for each role.
    - g. Click Save configuration to save your changes.

## Configuring the timeout length for the password reset link

---

You can configure the timeout length for the one-time password reset link that is sent from the Developer Portal to the admin user account.

## Before you begin

---

You must have administrator access to complete this task.

## About this task

---

By completing the steps in this task, you are configuring the timeout length for the one-time password reset link that is sent to the admin user account by the Developer Portal.

Note:

- Setting the password reset timeout length in the Developer Portal affects only the admin user account.
- You are not affecting the user activation or password reset links that are used by any other identification provider type in API Manager.

## Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the PEOPLE section, click Account settings.  
The Account settings window opens.
4. In the Account settings window, navigate to the PASSWORD RESET TIMEOUT section.
5. Enter your required value for the timeout in the Password Reset Timeout field.  
Note: The timeout value is specified in seconds, and is 86400 seconds by default.
6. Click Save configuration.  
You have configured the timeout length for the one-time link that is sent from the Developer Portal for the admin user account.

---

## Configuring your site password policy

You can configure your site password policy to define the constraints that are applied to the passwords of your Developer Portal users.

### Before you begin

---

You must have administrator access to complete this task.

### About this task

---

Any passwords that are stored in the API Manager must comply with the API Manager password policy. You can configure your password policy to be the same as or stricter than the API Manager password policy.

## Procedure

---

To configure your site password policy, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the SECURITY section, click Password Policy.  
The Password Policies page opens.
4. In the OPERATIONS column for the default password policy, click Edit.  
Note: You can create a new password policy by clicking Add Policy; this allows you to apply different password policy settings to different roles. If you need only a single password policy, just configure the pre-supplied default policy.
5. Click Next.  
The Configure Constraints page opens.
6. Configure the password constraints as required.
  - a. To configure a constraint, click Edit in the OPERATIONS column for the required constraint.
  - b. To delete a constraint, select Delete in the OPERATIONS column.
  - c. To add a constraint, select the required constraint type from the drop-down list, then click Configure Constraint Settings.
  - d. When you are done, click Next.  
The Apply to Roles page opens.
7. Select the roles to which the password policy settings are to be applied. then click Finish.

---

## Disabling CORS warnings

You can disable cross-origin resource sharing (CORS) warnings for unenforced APIs in the Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

### About this task

---

By disabling the CORS warnings, users will not receive further warnings regarding the usage of unenforced APIs.

## Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the SYSTEM section, click IBM API Connect.
4. Ensure that the Display CORS warnings for unenforced APIs check box is cleared.
5. Click Save Configuration.

You have disabled CORS warnings in the Developer Portal. To enable CORS warnings, repeat the steps and ensure that the Display CORS warnings for unenforced APIs check box is selected.

---

## Disabling live testing of APIs

You can disable live testing of APIs in the Developer Portal to restrict exposure of an API.

### Before you begin

---

You must have administrator access to complete this task.

### Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
  2. Click Configuration in the administrator dashboard.
  3. In the SYSTEM section, click IBM API Connect.
  4. Ensure that the Allow live testing of APIs check box is cleared.
  5. Click Save configuration.
- You have disabled live testing of APIs. To enable live testing of APIs, repeat the steps and ensure that the Allow live testing of APIs check box is selected.

---

## How to enable penetration testing, and information about Developer Portal cookies

Many security features are available in the Developer Portal, but if you need to do a penetration test of your Developer Portal site, some of these features might block the scan that is required as part of the testing. This topic provides guidance on the security features that you might need to disable to allow penetration testing of your site, as well as information about Developer Portal cookies.

Penetration testing, or pen testing, is the practice of testing a computer system, network, or web application for potential security vulnerabilities that an attacker might exploit. For certain elements of your pen testing, you might want to disable some of the security features that block scanning of the Developer Portal. The following sections describe the security features in the Developer Portal that you might need to disable. The actual requirements depend on your Developer Portal configuration, and on the type of pen testing that you are running. Note also the final section, which discusses testing for Distributed Denial of Service attacks.

Important:

- If you run a pen test, and you give the automated test tool the admin credentials, the tool submits all of the administration configuration forms as well, and this action is likely to break the Portal site configuration. Best practice is to use different credentials for the testing.
- If you disable security modules, and then run security scans, your Portal site configuration might be broken by the scanning process. Before running any security testing, ensure that you have a valid backup of your Portal site.
- If your Portal site configuration is broken by the security testing, then you must delete the Portal site from the API Manager Catalog settings and re-create it, or restore it from a backup.  
For information about restoring your site, see the appropriate topic for your environment:
  - [Backing up and restoring the Developer Portal in a Kubernetes environment](#)
  - [Backing up and restoring the Developer Portal on OpenShift and Cloud Pak for Integration](#)
  - [Backing up and restoring the Developer Portal in a VMware environment](#)
- Note that disabling of security modules makes it highly likely that you will find issues that are not actually vulnerabilities, because you have disabled the modules that are designed to protect against these vulnerabilities. Therefore, any issue that is found in these circumstances must be replicated in an environment without the security modules disabled, before contacting IBM Support.
- Do not use the **Enforced** mode of the **Content Security Policy**, this mode is not supported and can break core Developer Portal functionality such as API rendering and the content editor. The **Content Security Policy** must be in **Report only** mode.

Note: You might need to disable the **Honeypot** module to allow your penetration test to run. For more information, see [Using Honeypot for spam protection](#). This topic contains the following sections:

- [CAPTCHA and reCAPTCHA verifications](#)
- [Drupal Perimeter Defense module](#)
- [Login Security module](#)
- [Developer Portal cookies](#)
- [Should I test the Developer Portal for vulnerability to Distributed Denial of Service attacks?](#)

---

## CAPTCHA and reCAPTCHA verifications

CAPTCHA and reCAPTCHA verifications are a type of challenge-response test that are used in web applications to check whether the user is human. By default, your Developer Portal site is configured with an Image based CAPTCHA challenge enabled, but you can also configure a Math based CAPTCHA challenge, and Checkbox, Invisible, and Android reCAPTCHA challenges. If you need to disable these verifications before pen testing, see the following instructions.

To disable the CAPTCHA and reCAPTCHA verifications, you must disable the modules as follows:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. On the administrator dashboard, click **Extend > Disable** module.
4. Search through the list of modules, and select the check boxes next to the Image CAPTCHA and reCAPTCHA modules.
5. Click **Disable**, and click **Disable** again to confirm.
6. On the **Disable** tab, select the check box next to the CAPTCHA module, and click **Disable**. The display shows the forms that the CAPTCHA configuration is removed from.
7. Click **Disable** again to confirm. Note that the modules must be disabled in this order.

The CAPTCHA and reCAPTCHA modules are now disabled, and you can complete your pen testing. After pen testing, you can enable the modules again by clicking Extend on the administrator dashboard, selecting the check box for each module in the List tab, and clicking Enable.

## Drupal Perimeter Defense module

---

The Drupal Perimeter Defense module (machine name: perimeter) immediately bans the IP addresses of those users that send suspicious requests to your site, so any pen testing scans would be blocked. This module is enabled by default on the Developer Portal. For more information about the module, see the Drupal documentation [Drupal Perimeter Defense module](#).

If you need to disable the Drupal Perimeter Defense module before pen testing, see the following instructions:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. On the administrator dashboard, click Extend, Disable module.
4. Search through the list of modules, and select the check box next to the Drupal Perimeter Defense module.
5. Click Disable, and click Disable again to confirm.

The Drupal Perimeter Defense module is now disabled, and you can complete your pen testing. After pen testing, you can enable the module again by clicking Extend on the administrator dashboard, selecting the check box next to the Drupal Perimeter Defense module in the List tab, and clicking Enable.

## Login Security module

---

The Login Security module (machine name: login\_security) enables the site administrator to improve the security options in the login operation of your Developer Portal site. For example, controlling the number of invalid login attempts before blocking users, and denying access by IP address. This module is enabled by default on the Developer Portal. For more information about the module, see the Drupal documentation [Login Security module](#).

If you need to disable these security options before pen testing, you need to manually modify the configuration of your security options according to your requirements. For example, you might need to change the number of login attempts that are allowed before a user or IP address is blocked. To configure the security options, see the following instructions:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. On the administrator dashboard, click Configuration, People, Login Security.
4. In the Login Security window, update the security options according to your requirements.
5. Click Save configuration to save your updates.

The Login Security options are now updated, and you can complete your pen testing. After pen testing, you can reconfigure your security options in the Login Security window.

## Developer Portal cookies

---

Cookies are small text files that are placed on a computer by websites when they are visited. The Developer Portal uses cookies that remain on the computer for varying times. Some cookies expire at the end of each session, and some remain longer to enhance the user experience in the future. The list here explains the cookies that are used by the Developer Portal, and why they are used.

Note: Any load balancer that sits in front of the Developer Portal must be a transparent proxy that enables all cookies returned by the Developer Portal to pass through to the browser.

### Drupal session identifier cookie

This is the main cookie that all of the Developer Portal sites have, and it stores the session identification. The session cookie is named in the format `SSESS $UU$ UID`, where `UU` is a randomly generated unique string of characters for that session, and this is the session token itself. The session cookie is always marked with the following flags as instructions to the browser client:

- **Secure** - means that the cookie is sent only over a secure connection (that is, over TLS).
- **HttpOnly** - means that the browser adds this cookie on outbound requests only, and not provide the value to client side javascript.

### session\_store.id cookie

This cookie is used to store information on where a user is in the subscription flow, and any options that they selected, if they haven't yet signed in, so the Developer Portal can redirect them back there later. The session\_store.id cookie is also marked with the **Secure** and **HttpOnly** flags, and it does not contain any confidential information.

### Drupal.visitor.startSubscriptionWizard cookie

This cookie is used to store information on where a user started in the subscription flow if they haven't yet signed in, so the Developer Portal can redirect them back there later. The Drupal.visitor.startSubscriptionWizard cookie is also marked with the **Secure** and **HttpOnly** flags, and it does not contain any confidential information.

### portal\_web cookie

This cookie is used to achieve session persistence, which it does by ensuring that web traffic between the user and the Developer Portal is correctly routed to the server. The portal\_web cookie is used for both HTTP and HTTPS communications, so that routing is meaningful irrespective of the transfer protocol, and ensuring consistent behavior if the switching of protocols is required. The cookie contains only a hash of the pod that served the traffic to maintain session persistence, and does not contain any confidential information. The cookie is provided by the Portal ingress. This cookie is not marked as **Secure** or **HttpOnly**, as the Kubernetes ingress does not yet support these attributes.

### Other cookies

In addition to the cookies listed previously, there are other (usually temporary) cookies that might be created depending on the configuration options of your Developer Portal site. For example, if the site uses OAuth for authentication, then during the OAuth token activation flow a JSON Web Token is sent to the browser. This token is stored in a cookie for a limited lifetime. This action is part of the normal function of an OAuth flow. As the creation of these other cookies depends on your particular Developer Portal configuration, we cannot provide a complete list of cookies here.

## Should I test the Developer Portal for vulnerability to Distributed Denial of Service attacks?

---

When you run penetration testing, it's important to understand that individual systems are not responsible for (nor capable of dealing with) Distributed Denial of Service (DDoS) attacks. DDoS attacks are launched by huge numbers of external computers simultaneously, with a goal of using up all of the resources of a system, and thereby

denying service to legitimate users. These kinds of attacks are usually defended against at the level of the network and edge gateways, and information about how to protect against them is out of the scope of this topic. Although the Developer Portal does have some capabilities, including the modules mentioned previously, for potentially mitigating some of the effects of a DDoS attack, do not rely on these capabilities as the first line of defense; rather they are designed to assist in dealing only with any elements of an attack that might have gotten through the outer network layers of protection.

---

## Related tasks

- [Configuring CAPTCHA](#)
- [Configuring reCAPTCHA](#)
- [Disabling modules](#)
- [Login security](#)
- [Blocking and unblocking specific users](#)
- [Managing banned IP addresses](#)

---

## Related information

- [Understanding Drupal](#)

---

# How to manage IP security in the Developer Portal

The Developer Portal offers the ability to perform various IP address security measures, such as adding and removing specific IP addresses from the banned IP address list, automatically banning client IP addresses by using the Drupal Perimeter Defence module, or managing login security by using flood control.

See the following links for information about how to manage IP security in the Developer Portal:

- [Enable and disable IP security](#)
- [Correctly passing through client IP addresses](#)
- [Managing banned IP addresses](#)

---

## Enable and disable IP security

The ability to enable or disable IP security related actions can be controlled at the Portal service level by using the Developer Portal CLI command `ip-security-enabled`. When IP security is enabled, modules such as the Drupal Perimeter Defence module, or flood control, will block client IP addresses suspected of malicious behavior, as expected. When IP security is disabled, all IP related security is switched off. For example, the perimeter module won't block client IP addresses when IP security is disabled. You might want to turn off IP security if you are performing penetration tests, or if you cannot pass through the client IP address from your external load balancer. For more information, see [Using the ip-security-enabled command](#).

---

## Correctly passing through client IP addresses

In order to correctly use modules that make use of client IP banning, such as the Drupal Perimeter Defence module, you must ensure that any external load balancer that fronts the Portal cluster passes through the client IP address. This can be achieved by passing the client IP address through in an `'x-forwarded-for'` header, or by making use of the proxy protocol (provided both the load balancer and the ingress controller are compatible with the proxy protocol, and have the protocol enabled). Failure to correctly pass through the client IP address, results in the load balancer IP address being blocked when a client attempts to send a suspicious request to the portal.

If you are fronting the Portal cluster with an HAProxy setup acting as the load balancer, then you can make use of the proxy protocol by adding the `send-proxy` directive to the end of the Portal server declarations in the HAProxy configuration file, for example:

```
server portal0 portal_host:port check send-proxy
```

You must restart the HAProxy for the change to take effect. Note that if your load balancer IP address has already been blocked, you will need to remove the blocked IP address from the banned list by using the `security` command in the Developer Portal CLI, see [Using the security command](#).

If you find your load balancers are being banned by Portal's IP Security modules, then you might need to configure your system's ingresses to send the `'x-forwarded-for'` headers. For instance, the default behavior of the `nginx-ingress-controller` is to ignore the inbound `'x-forwarded-for'` header and construct a new one. To configure the `nginx-ingress-controller` to pass through the inbound `'x-forwarded-for'` header you need to add the following to the `nginx-ingress-controller` config map:

```
data:
 compute-full-forwarded-for: "true"
 use-forwarded-headers: "true"
```

For more information about the Drupal Perimeter Defence module, see [Drupal Perimeter Defense module](#).

---

## Managing banned IP addresses

You can manage banned IP addresses for a particular site by using the administrator dashboard for a particular Developer Portal site. For more information, see [Managing banned IP addresses](#).

---

## Related concepts

- [How to enable penetration testing, and information about Developer Portal cookies](#)

---

## Related tasks

- [Disabling modules](#)
- [Login security](#)
- [Blocking and unblocking specific users](#)
- [Managing banned IP addresses](#)

---

## Managing banned IP addresses

You can ban specific IP addresses from accessing your Developer Portal site.

---

### Before you begin

You must have administrator access to complete this task.

---

### Procedure

To manage banned IP addresses, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the PEOPLE section, click IP address bans.  
The IP address bans window opens.
4. You can complete the following tasks:
  - a. To ban an IP address, enter the address in the IP address field, then click Add.  
The address is added to the BANNED IP ADDRESSES list.
  - b. To remove an IP address from the banned list, click Delete alongside the required address.

Note:

- There is automatic login "flood" protection built into the Developer Portal. This feature means that an excess of login attempts from the same IP address in one hour will cause that IP to be temporarily banned. For more information, see [Using flood control for login security](#).
- You can also clear banned user/IP addresses by using the Developer Portal CLI command `security:clear-bans`. For more information, see [Using the security command](#).

---

## Using flood control for login security

You can configure login security for your Developer Portal by using flood control.

---

### Before you begin

You must have administrator access to complete this task.

---

### Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Navigate to Configuration > System > Flood Control.
3. Enter values in the following fields in the LOGIN SETTINGS section according to your requirements:
  - Failed IP login limit (min 1) - the number of failed login attempts from the same client IP.
  - Failed IP login window in seconds (0 = Off) - the time window, in seconds, for the number of failed login attempts from the same client IP.
  - Failed User login limit (min 1) - the number of failed login attempts from a user.
  - Failed User login window in seconds (0 = Off) - the time window, in seconds, for the number of failed login attempts from a user.
4. Enter values in the following fields in the CONTACT FORM section according to your requirements:
  - Emails sent limit - the number of emails that can be sent.
  - Emails sent window in seconds - the time window, in seconds, for the number of emails that can be sent.
5. Click Save configuration.

---

### What to do next

To unblock a user, see [Blocking and unblocking specific users](#). To unblock an IP address, see [Managing banned IP addresses](#).

---

## Login security

You can configure the login security for your Developer Portal at an IP and user level. You can also configure login security for Developer Portal contact forms.

---

### Before you begin

Important: The Login Security module is being removed from future releases. It has been replaced by Flood Control. Existing users of the Login security module should configure Flood Control with the same limits and time windows as specified here. For more information, see [Using flood control for login security](#). You must have administrator access to complete this task.



## Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the PEOPLE section, click Login Security.
4. Enter values in the following fields in the GENERAL SETTINGS section according to your requirements:
  - Track time - the time window to check for security violations.
  - User - the number of login failures before blocking a user.
  - Soft host - the number of login failures before soft blocking a host. The host can still browse the site as an anonymous user.
  - Host - the number of login failures before hard blocking a host. The host cannot access the site at all.
  - Attack detection - the number of login failures before warning about an ongoing attack.
5. Select the check boxes that satisfy your notification requirements in the NOTIFICATION section:
  - Disable login failure error message
  - Notify the user about the number of remaining login attempts
  - Display last login timestamp
  - Display last access timestamp
6. Optional: Edit the EMAIL FOR ONGOING ATTACK DETECTION section.
7. Optional: Edit the EMAIL FOR BLOCKED ACCOUNT section.
8. Optional: You can edit the notification texts that are displayed by configuring the following options:
  - Failed login attempt
  - Banned host (soft IP ban)
  - Banned host (hard IP ban)
  - Blocked user by uid
9. After you have configured your login security options, click Save configuration.

## What to do next

---

To unblock a user, see [Blocking and unblocking specific users](#). To unblock an IP address, see [Managing banned IP addresses](#).

## Using the Security kit

You can improve the security of your website by configuring various options that are available in the Security Kit module, in the Developer Portal.

## Before you begin

---

You must have administrator access to complete this task.

## About this task

---

The Security Kit module provides your website with various options to mitigate risks of common web application vulnerabilities like Cross-site Scripting, Cross-site Request Forgery and Clickjacking. It also has some options to improve your SSL/TLS security.

## Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the SYSTEM section, click Security Kit settings.
4. In the Security Kit module, configure the security settings for your website as required.
5. When you are done, click Save configuration.

## Using Honeypot for spam protection

Honeypot protection provides security mechanisms to protect your Developer Portal site from form submission by spam bots. If spam bot activity is detected, form submission is blocked.

## Before you begin

---

You must have administrator access to complete this task.

## About this task

---

Honeypot protection provides the following security mechanisms:

- A hidden field, unseen by users, is added to the form. If a value has been entered in the field when the form is submitted then this indicates that the form was completed by a spam bot, and the submission is blocked. You can specify the name of the hidden field.
- If the form is submitted before a specified time has elapsed (five seconds by default), it is assumed that this is too short a time for a human to have completed the form, and the submission is blocked. You can specify the time length.

Honeypot protection is provided by the Honeypot module, which is enabled by default.

Note: If you want to carry out automated testing of your Developer Portal, you might need to disable the Honeypot module, because Honeypot spam protection is designed specifically to block automated Developer Portal usage. For details on how to disable a module, see [Disabling modules](#).

## Procedure

---

To configure Honeypot for spam protection in the Developer Portal, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the CONTENT AUTHORING section, click Honeypot configuration.
4. Specify the forms that you want to protect with Honeypot.
  - a. To enable Honeypot protection for all the forms on your Developer Portal site, select Protect all forms with Honeypot.
  - b. To choose which forms you want to protect with Honeypot, clear the Protect all forms with Honeypot check box, then select the required forms in the HONEYPOT ENABLED FORMS section.By default, Honeypot protection is enabled for all user management forms and all comment forms.
5. To have details of all blocked form submissions written to the log file, select Log blocked form submissions.
6. In the Honeypot element name field, specify the name of the hidden form field.

The default field name is url. You need to change the value if your form already has a field of the same name. For the most effective protection, use a generic field name; for example, email, homepage, or link.
7. In the Honeypot time limit field, specify the number of seconds that must elapse before it is assumed that a form is being submitted by a human rather than a spam bot. If the form is submitted before this time has elapsed then the submission is blocked.

The default value is five seconds.
8. When done, click Save configuration.

## People

---

You can manage and customize the user experience in the Developer Portal, by altering permissions and assigning roles to specific users.

To learn how to configure the user experience for your Developer Portal site, use the following section links.

- [Blocking and unblocking specific users](#)  
You can block and unblock specific users in the Developer Portal.
- [Controlling access to Developer Portal content](#)  
In IBM® API Connect, you can restrict access to content for Consumer organizations through an access content list. However, you can also assign permissions in the Developer Portal that override the access content list, and allow users to access and edit all of the content for Products, APIs, and Applications.
- [Creating a developer account to customize API properties](#)  
You can allow specific users in the Developer Portal to customize the properties for certain APIs, including APIs that are not viewable by the administrator, or other customizable features, by creating an internal Consumer organization.
- [Email Subscribers](#)  
As a site administrator, you can use the email subscribers wizard to send messages to your Developer Portal community.
- [How to style the Developer Portal emails](#)  
You can apply a theme to the HTML emails that are sent by the Developer Portal by using CSS and templating.
- [Working with roles in the Developer Portal](#)  
As a site administrator, you can create and customize site configuration roles in the Developer Portal.

## Blocking and unblocking specific users

---

You can block and unblock specific users in the Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

## Procedure

---

To block or unblock a specific user, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click People in the administrator dashboard.

A list of users is displayed.
3. Select the check box next to the target user.
4. From the drop-down list under Action, select Block the selected user(s) or Unblock the selected user(s).
5. Click Apply to selected items.

You have blocked or unblocked specific users from the Developer Portal.

Note: If a user reports that they are unable to login, but they do not appear as blocked, then it might be that they have been blocked by Flood Control. To unblock such users, select Configuration > System > Flood unblock. You can configure your Flood Control settings; see [Using flood control for login security](#) for more information.

## Related tasks

---

- [Using flood control for login security](#)

---

## Controlling access to Developer Portal content

In IBM® API Connect, you can restrict access to content for Consumer organizations through an access content list. However, you can also assign permissions in the Developer Portal that override the access content list, and allow users to access and edit all of the content for Products, APIs, and Applications.

---

### Before you begin

You must have administrator access to complete this task.

---

### About this task

A user with a Content Author role is a curator of content for the Developer Portal. A Content Author user can perform the following actions:

- Set taxonomies on the content
- Provide custom icons for content
- Upload additional documentation files
- Create documentation pages in the Developer Portal.

By default, Content Author users are assigned the following user permissions:

Edit any Product content

The user can access and edit all of the Products available in the Catalog that the Developer Portal is associated with. The user can also access APIs that belong to those Products.

Edit any API content

The user can access and edit all of the APIs that are available in the Catalog, but not the Products with which they are associated. (Note that this permission is a subset of Edit any Product content; as Content Author users are assigned both permissions they can access and edit all of the Products and the associated APIs that are available.)

The Edit any Application permission grants the user access to all of the Applications that are in the Developer Portal, but is not assigned to any user or role by default.

Note: There are security implications if you grant the Edit any Application permission, and it should be assigned only to trusted roles.

A user can configure, attach, and upload content if they are assigned the appropriate permissions.

---

### Procedure

To check and assign permissions that grant users access to all content, proceed with one of the following options:

- Assign a user to the Content Author role. For more information, see [Assigning users to a role](#).
- Create a custom role, to which you can add the relevant permissions and add users to your new custom role. For more information, see [Creating a new role](#), [Assigning permissions to a role](#), and [Assigning users to a role](#).

---

## Creating a developer account to customize API properties

You can allow specific users in the Developer Portal to customize the properties for certain APIs, including APIs that are not viewable by the administrator, or other customizable features, by creating an internal Consumer organization.

---

### Before you begin

You must have administrator access to complete this task.

---

### About this task

Every user in the Developer Portal must be a member of a Consumer organization. To enable additional users to be editors of particular content in the Developer Portal, you can create an internal Consumer organization. You can then invite these additional editors to this organization, and assign them a specific role that grants certain editing permissions. For example, the following API properties can be customized by any member of the new Consumer organization that have been assigned the Administrator or Content Author role:

- Image uploading
- Tag assignment
- File attachment
- Custom field editing

---

### Procedure

To create a new Consumer organization within the Developer Portal, complete the following steps:

1. Log in to the Developer Portal as an administrator or Content Author.
2. Click the arrow next to `your_organization_name` from the Developer Portal home page, and select Create organization.
3. In the Organization Title field, type the name of the organization that you want, then click Submit.

To invite the editor users to your new organization, complete the following steps:

4. Switch to your newly created organization by clicking the arrow next to `your_current_organization_name`, and select your new organization name from the list.
5. Click the arrow next to `your_new_organization_name` and select My organization.

Your new organization page is displayed.

6. For each new user, click Invite, complete their Email details, and click Submit. Note that you can leave the default role of Developer selected and then change the role later, or you can select the Administrator role now. However, be aware that an administrator is able to perform any task within the Developer Portal that does not involve the creation of APIs, Products, and Apps. You should be careful to ensure that only trusted users are given this access and level of control of your site. Each new user will receive an invitation email, and they need to click on the link in the email to activate their Developer Portal account.

To assign the required permissions to the users of your organization, you must log in as an administrator and complete the following steps:

7. Log in to the Developer Portal as an administrator.
8. If the administrator dashboard is not displayed, click Manage to display it.
9. Click People on the administrator dashboard.
10. Select the check box for the user, or users, to whom you want to give new permissions.
11. Under the Action heading, select the relevant role from the drop-down list.  
For example, selecting Add the Content Author role to the selected user(s) means they can edit images and content within the Developer Portal.
12. Click Apply to selected items to update the roles.

---

## What to do next

If you want specific Products or APIs to be customized by the editor users, you must ensure that the relevant Products are visible to the internal Consumer organization that you have created. For more information on how to make a plan visible in API Manager, see [Changing the availability of a Product](#). Note that you can also create new roles and assign specific permissions to those roles. For more information, see [Creating a new role](#) and [Assigning permissions to a role](#).

---

## Related concepts

- [Working with roles in the Developer Portal](#)

---

## Email Subscribers

As a site administrator, you can use the email subscribers wizard to send messages to your Developer Portal community.

You can email all the subscribers of a specific API, plan, product, or all registered consumer organizations. For each consumer organization, you can select whether to email just the owner or all of the members. Each recipient receives an individual email.

You can use the wizard to specify context-specific tokens in the email subject and body. The tokens available and their replacement values depend on which flow is selected, for example, API, plan, product, or all.

- [Emailing all subscribers](#)  
As a site administrator, you can use the email **All users** wizard to email all the subscribers of all the consumer organizations on your Developer Portal.
- [Emailing api subscribers](#)  
As a site administrator, you can use the email **API subscribers** wizard to send messages to all subscribers of all product plans that contain a specific API on your Developer Portal.
- [Emailing plan subscribers](#)  
As a site administrator, you can use the email **Plan subscribers** wizard to send messages to all subscribers of a chosen plan that belongs to a specific product on your Developer Portal.
- [Emailing product subscribers](#)  
As a site administrator, you can use the email **Product subscribers** wizard to send messages to all subscribers of any plan for a specific product in your Developer Portal.
- [Token reference](#)  
You can use the email subscribers wizard to specify context-specific tokens in the email subject and body. The tokens available and their replacement values depend on which flow is selected, for example, API, plan, product, or all.

---

## Emailing all subscribers

As a site administrator, you can use the email **All users** wizard to email all the subscribers of all the consumer organizations on your Developer Portal.

---

## Before you begin

You must have administrator access to complete this task.

---

## About this task

You can use the **All users** wizard to send messages to all consumer organizations. You can select whether to email just the owner, or all of the members of the consumer organizations. Each recipient receives an individual email.

---

## Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Navigate to People, Email subscribers.
3. Select **All users**, and click Save.
4. Select the role of the recipients that you want to send the email to, then click Next.
  - Owners - sends the email to the relevant consumer organization owners.
  - Members - sends the email to all members of the relevant consumer organizations.

5. Enter the content of the email, subject, and body, and set the Text Format of the email:
  - Basic HTML
  - Plain text
6. Expand the REPLACEMENTS drop-down, and click Browse available tokens to view which tokens can be added to the subject and body fields for replacement. You can select a token to use by either typing it directly, or by clicking in the subject or body field. You are then given a list of available tokens, and you can insert the token that you want. For a reference list of tokens, see [Token reference](#).
7. Expand the ADDITIONAL EMAIL OPTIONS drop-down to configure any additional settings:
  - **Priority:** Set the priority of the email. Many email programs ignore a priority setting.
  - **Request receipt:** Request a read receipt from your emails. Many email programs ignore these requests so it is not a definitive indication of how many people read your email.
  - **Additional headers:** Set any additional headers to be sent with the email.
8. Use the remaining check boxes to configure what messages are to be sent and how:
  - Send the message directly by using the Batch API. This option sends the emails directly by the Batch API. If not selected, then the emails are added to the spool and sent out by cron.
  - Send a copy of all messages to the sender. Because emails might contain context-specific tokens, a new email is generated for each recipient with the correct token replacement. This option sends a copy of all generated emails to the sender.
  - Send a copy of the original message to the sender. This option sends a single copy of the original email, subject, and body, without the replacement of any specified tokens.
9. Click Next.
10. Confirm that the email is ready to send, then click Next.
11. If you chose to send the emails by the Batch API, you see a progress page that indicates how many emails were sent. You are then redirected to a summary page that shows the total number of emails that were sent. If you chose to spool the delivery of the emails, you are redirected to the summary page that shows the total number of emails that are spooled.
12. Click Finish.

---

## Results

You successfully sent an email to all members, or all of the owners of each consumer organization on your Developer Portal.

---

## Emailing api subscribers

As a site administrator, you can use the email **API subscribers** wizard to send messages to all subscribers of all product plans that contain a specific API on your Developer Portal.

---

## Before you begin

You must have administrator access to complete this task.

---

## About this task

You can use the email **API subscribers** wizard to send messages to all subscribers of all product plans that contain a specific API in your Developer Portal community. For each consumer org, you can select whether to email just the owner or all members. Each recipient receives an individual email.

---

## Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Navigate to People > Email subscribers.
3. Select **API subscribers**, and click Save.
4. In the text box, enter the name of the API whose subscribers you want to email. If you start typing in the text box, you can choose from the options that you are given. Click Next.
5. Select the role of the recipients that you want to send the email to, then click Next.
  - Owners - sends the email to the relevant consumer organization owners.
  - Members - sends the email to all members of the relevant consumer organizations.
6. Enter the content of the email, subject, and body, and set the Text Format of the email:
  - Basic HTML
  - Plain text
7. Expand the REPLACEMENTS drop-down, and click Browse available tokens to view which tokens can be added to the subject and body fields for replacement. You can select a token to use by either typing it directly, or by clicking in the subject or body field. You are then given a list of available tokens, and you can insert the token that you want. For a reference list of tokens, see [Token reference](#).
8. Expand the ADDITIONAL EMAIL OPTIONS drop-down to configure any additional settings:
  - **Priority:** Set the priority of the email. Many email programs ignore a priority setting.
  - **Request receipt:** Request a read receipt from your emails. Many email programs ignore these requests so it is not a definitive indication of how many people read your email.
  - **Additional headers:** Set any additional headers to be sent with the email.
9. Use the remaining check boxes to configure what messages are to be sent and how:
  - Send the message directly by using the Batch API. This option sends the emails directly by the Batch API. If not selected, then the emails are added to the spool and sent out by cron.
  - Send a copy of all messages to the sender. Because emails might contain context-specific tokens, a new email is generated for each recipient with the correct token replacement. This option sends a copy of all generated emails to the sender.
  - Send a copy of the original message to the sender. This option sends a single copy of the original email, subject, and body, without the replacement of any specified tokens.
10. Click Next.

11. Confirm that the email is ready to send, then click Next.
12. If you chose to send the emails by the Batch API, you see a progress page that indicates how many emails were sent. You are then redirected to a summary page that shows the total number of emails that were sent.  
If you chose to spool the delivery of the emails, you are redirected to the summary page that shows the total number of emails that are spooled.
13. Click Finish.

---

## Results

You successfully sent an email to all members or all owners of each consumer org that contained a subscription to all product plans that contain the chosen API on your Developer Portal.

---

## Emailing plan subscribers

As a site administrator, you can use the email **Plan subscribers** wizard to send messages to all subscribers of a chosen plan that belongs to a specific product on your Developer Portal.

---

## Before you begin

You must have administrator access to complete this task.

---

## About this task

You can use the email **Plan subscribers** wizard to send messages to all subscribers of a chosen plan that belongs to a specific product on your Developer Portal community. For each consumer org, you can select whether to email just the owner or all members. Each recipient receives an individual email.

---

## Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Navigate to People > Email subscribers.
3. Select **Plan subscribers**, and click Save.
4. In the text box, enter the name of the product whose subscribers you want to email. If you start typing in the text box, you can choose from the options that you are given. Click Next.
5. Select the name of the plan whose subscribers you want to email. Click Next.
6. Select the role of the recipients that you want to send the email to, then click Next.
  - Owners - sends the email to the relevant consumer organization owners.
  - Members - sends the email to all members of the relevant consumer organizations.
7. Enter the content of the email, subject, and body, and set the Text Format of the email:
  - Basic HTML
  - Plain text
8. Expand the REPLACEMENTS drop-down, and click Browse available tokens to view which tokens can be added to the subject and body fields for replacement. You can select a token to use by either typing it directly, or by clicking in the subject or body field. You are then given a list of available tokens, and you can insert the token that you want. For a reference list of tokens, see [Token reference](#).
9. Expand the ADDITIONAL EMAIL OPTIONS drop-down to configure any additional settings:
  - **Priority**: Set the priority of the email. Many email programs ignore a priority setting.
  - **Request receipt**: Request a read receipt from your emails. Many email programs ignore these requests so it is not a definitive indication of how many people read your email.
  - **Additional headers**: Set any additional headers to be sent with the email.
10. Use the remaining check boxes to configure what messages are to be sent and how:
  - Send the message directly by using the Batch API. This option sends the emails directly by the Batch API. If not selected, then the emails are added to the spool and sent out by cron.
  - Send a copy of all messages to the sender. Because emails might contain context-specific tokens, a new email is generated for each recipient with the correct token replacement. This option sends a copy of all generated emails to the sender.
  - Send a copy of the original message to the sender. This option sends a single copy of the original email, subject, and body, without the replacement of any specified tokens.
11. Click Next.
12. Confirm that the email is ready to send, then click Next.
13. If you chose to send the emails by the Batch API, you see a progress page that indicates how many emails were sent. You are then redirected to a summary page that shows the total number of emails that were sent.  
If you chose to spool the delivery of the emails, you are redirected to the summary page that shows the total number of emails that are spooled.
14. Click Finish.

---

## Results

You successfully sent an email to all members or all owners of each consumer org that contained a subscription to the chosen plan within the chosen product, on your Developer Portal.

---

## Emailing product subscribers

As a site administrator, you can use the email **Product subscribers** wizard to send messages to all subscribers of any plan for a specific product in your Developer Portal.

## Before you begin

You must have administrator access to complete this task.

## About this task

You can use the email **Product subscribers** wizard to send messages to all subscribers of any plan for a specific product in your Developer Portal community. For each consumer org, you can select whether to email just the owner or all members. Each recipient receives an individual email.

## Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Navigate to People > Email subscribers.
3. Select **Product subscribers**, and click Save.
4. In the text box, enter the name of the product whose subscribers you want to email. If you start typing in the text box, you can choose from the options that you are given. Click Next.
5. Select the role of the recipients that you want to send the email to, then click Next.
  - Owners - sends the email to the relevant consumer organization owners.
  - Members - sends the email to all members of the relevant consumer organizations.
6. Enter the content of the email, subject, and body, and set the Text Format of the email:
  - Basic HTML
  - Plain text
7. Expand the REPLACEMENTS drop-down, and click Browse available tokens to view which tokens can be added to the subject and body fields for replacement. You can select a token to use by either typing it directly, or by clicking in the subject or body field. You are then given a list of available tokens, and you can insert the token that you want. For a reference list of tokens, see [Token reference](#).
8. Expand the ADDITIONAL EMAIL OPTIONS drop-down to configure any additional settings:
  - **Priority**: Set the priority of the email. Many email programs ignore a priority setting.
  - **Request receipt**: Request a read receipt from your emails. Many email programs ignore these requests so it is not a definitive indication of how many people read your email.
  - **Additional headers**: Set any additional headers to be sent with the email.
9. Use the remaining check boxes to configure what messages are to be sent and how:
  - Send the message directly by using the Batch API. This option sends the emails directly by the Batch API. If not selected, then the emails are added to the spool and sent out by cron.
  - Send a copy of all messages to the sender. Because emails might contain context-specific tokens, a new email is generated for each recipient with the correct token replacement. This option sends a copy of all generated emails to the sender.
  - Send a copy of the original message to the sender. This option sends a single copy of the original email, subject, and body, without the replacement of any specified tokens.
10. Click Next.
11. Confirm that the email is ready to send, then click Next.
12. If you chose to send the emails by the Batch API, you see a progress page that indicates how many emails were sent. You are then redirected to a summary page that shows the total number of emails that were sent. If you chose to spool the delivery of the emails, you are redirected to the summary page that shows the total number of emails that are spooled.
13. Click Finish.

## Results

You successfully sent an email to all members or all owners of each consumer org that contained a subscription to any plan within the chosen product on your Developer Portal.

## Token reference

You can use the email subscribers wizard to specify context-specific tokens in the email subject and body. The tokens available and their replacement values depend on which flow is selected, for example, API, plan, product, or all.

## Application Tokens

Tokens that are related to an individual Application.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Description	[application:description]	The description that is given to the application.	✓	✓	✓	
Enabled	[application:enabled]	The state that the application is in.	✓	✓	✓	
ID	[application:id]	The ID of the application.	✓	✓	✓	
Image URL	[application:image_url]	The URL of the image for the application.	✓	✓	✓	
Lifecycle State	[application:lifecycle_state]	The lifecycle state of the application.	✓	✓	✓	
Name	[application:name]	The name of the application.	✓	✓	✓	

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Title	[application:title]	The title of the application.	✓	✓	✓	

## Consumer Organization Tokens

Tokens that are related to the individual consumer organization.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
ID	[consumer-org:id]	The ID of the consumer organization.	✓	✓	✓	✓
Name	[consumer-org:name]	The name of the consumer organization.	✓	✓	✓	✓
Title	[consumer-org:title]	The title of the consumer organization.	✓	✓	✓	✓

## Product Tokens

Tokens that are related to an individual product.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
APIs	[product:apis]	A comma separated list of APIs (name:version) within the product.	✓	✓	✓	
ID	[product:id]	The ID of the product.	✓	✓	✓	
Image URL	[product:image_url]	The URL of the image for the product.	✓	✓	✓	
Name	[product:name]	The name of the product.	✓	✓	✓	
Plans	[product:plans]	A comma separated list of plan names within the product.	✓	✓	✓	
State	[product:state]	The state of the product.	✓	✓	✓	
Title	[product:title]	The title of the product.	✓	✓	✓	

## Plan Tokens

Tokens that are related to an individual product plan.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Approval required	[product-plan:approval]	Whether the plan requires approval or not.		✓		
Description	[product-plan:description]	The description of the plan.		✓		
Name	[product-plan:name]	The name of a given plan within the product.		✓		
Rate limits	[product-plan:rate-limits]	A comma separated list of rate limit name value pairs that the plan contains.		✓		
Title	[product-plan:title]	The title of the plan.		✓		

## API Tokens

Tokens that are related to an individual API.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
ID	[api:id]	The ID of the API.			✓	
Image URL	[api:image_url]	The URL of the image for the API.			✓	
Name	[api:name]	The name of the API.			✓	
OAI Version	[api:oai_version]	The OAI version of the API.			✓	
Protocol	[api:protocol]	The protocol of the API.			✓	
State	[api:state]	The current state the API is in.			✓	
Title	[api:title]	The title of the API.			✓	
Version	[api:version]	The version of the API.			✓	

## User Tokens

Tokens that are related to the individual user account of the recipient.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Account Name	[user:account-name]	The login name of the user account.	✓	✓	✓	✓
Created	[user:created]	The date the user account was created.	✓	✓	✓	✓
Display Name	[user:display-name]	The display name of the user account.	✓	✓	✓	✓
Email	[user:mail]	The email address of the user account.	✓	✓	✓	✓
First Name	[user:first_name]	The first name of the user.	✓	✓	✓	✓



Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Last Name	[user:last_name]	The last name of the user.	✓	✓	✓	✓
Roles	[user:roles]	The user roles associated with the user account.	✓	✓	✓	✓
State	[user:apic_state]	List (text) field.	✓	✓	✓	✓
User ID	[user:uid]	The unique ID of the user account.	✓	✓	✓	✓

## Current User Tokens

Tokens that are related to the current user who is initiating the sending of the emails.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Account name	[current-user:account-name]	The login name of the user account.	✓	✓	✓	✓
Display name	[current-user:display-name]	The display name of the user account.	✓	✓	✓	✓
Email	[current-user:mail]	The email address of the user account.	✓	✓	✓	✓

## Site Information Tokens

Tokens that are related to site-wide settings and other global configuration.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Email	[site:mail]	The administrative email address for the site.	✓	✓	✓	✓
Login page	[site:login-url]	The URL of the site's login page.	✓	✓	✓	✓
Name	[site:name]	The name of the site.	✓	✓	✓	✓
Slogan	[site:slogan]	The slogan of the site.	✓	✓	✓	✓
URL	[site:url]	The URL of the site's front page.	✓	✓	✓	✓
URL (brief)	[site:url-brief]	The URL of the site's front page without the protocol.	✓	✓	✓	✓

## Current Date Tokens

Tokens that are related to the current date and time.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Custom format	[current-date:custom:?]	A date in a custom format. See the PHP documentation for details.	✓	✓	✓	✓
Fallback date format	[current-date:fallback]	A date in 'fallback' format. (Fri, 01/08/2021 - 14:19)	✓	✓	✓	✓
HTML Date	[current-date:html_date]	A date in 'html_date' format. (2021-01-08)	✓	✓	✓	✓
HTML Datetime	[current-date:html_datetime]	A date in 'html_datetime' format. (2021-01-08T14:19:19+0000)	✓	✓	✓	✓
HTML Month	[current-date:html_month]	A date in 'html_month' format. (2021-01)	✓	✓	✓	✓
HTML Time	[current-date:html_time]	A date in 'html_time' format. (14:19:19)	✓	✓	✓	✓
HTML Week	[current-date:html_week]	A date in 'html_week' format. (2021-W01)	✓	✓	✓	✓
HTML Year	[current-date:html_year]	A date in 'html_year' format. (2021)	✓	✓	✓	✓
HTML Yearless date	[current-date:html_yearless_date]	A date in 'html_yearless_date' format. (01-08)	✓	✓	✓	✓
Long format	[current-date:long]	A date in 'long' format. (Friday, January 8, 2021 - 14:19)	✓	✓	✓	✓
Medium format	[current-date:medium]	A date in 'medium' format. (Fri, 01/08/2021 - 14:19)	✓	✓	✓	✓
Raw timestamp	[current-date:raw]	A date in UNIX timestamp format (1610115559)	✓	✓	✓	✓
Short format	[current-date:short]	A date in 'short' format. (01/08/2021 - 14:19)	✓	✓	✓	✓
Time-since	[current-date:since]	A date in 'time-since' format. (6 minutes)	✓	✓	✓	✓

## How to style the Developer Portal emails

You can apply a theme to the HTML emails that are sent by the Developer Portal by using CSS and templating.

## Before you begin

---

You must have administrator access to complete this task.

You must have a custom theme. For information about how to create a custom theme, see [Creating a sub-theme](#).

## About this task

---

You can apply custom styling to the HTML content of the emails that are sent by the Developer Portal, such as the emails that are sent by the Email subscribers wizard. The custom styling consists of two parts: editing a twig template and, optionally, also editing a CSS file in your custom theme. See the following instructions for details. Note: These instructions apply only to emails that are sent by the Developer Portal. These instructions don't apply to the consumer invitation, activation, and password reset emails, as these emails are sent by the API Manager. For information about how to style API Manager emails, see [Configuring notifications](#) and [Sending messages to consumer organization owners](#).

## Procedure

---

1. Download the `ibm-apic-mail-message.html.twig` template from [https://github.com/ibm-apiconnect/devportal/tree/master/ibm\\_apim/ibm\\_apic\\_mail/templates](https://github.com/ibm-apiconnect/devportal/tree/master/ibm_apim/ibm_apic_mail/templates), and save it to your custom theme templates directory.
2. Edit the `ibm-apic-mail-message.html.twig` template to fit your custom theme.  
You can add styles directly into this twig template, or you can add them to the `css/mail.css` file in your custom theme, which will mean that they are automatically included in the HTML email content that is sent.

If you are using an SCSS theme that has been generated by the Developer Portal theme generator, then you can put the custom SCSS mail styles into the `mail-overrides.scss` file. These styles will then be compiled into the `mail.css` file.

Note: If you are using an older theme that was generated before API Connect Version 10.0.4.0, then the theme won't contain the mail theme support. However, you can workaround this by generating a new sub-theme with the same name, copying the `overrides.css`, or `overrides.scss`, as appropriate, `logo.png`, and `*.theme` files from the old theme into the new sub-theme, and then compress and install the new sub-theme as usual. (See [Creating a sub-theme](#) for details.)

3. Define a standard set of classes in the `ibm-apic-mail-message.html.twig` template that the Developer Portal administrators can use.  
The twig template lists the variables that can be used within it. The HTML `<body>` element is the actual content of the email message as defined within Drupal. For example, that content could be the information that's entered in the WYSIWYG editor of the Email subscribers wizard. Note that the HTML content in the WYSIWYG editor is more restricted for security reasons, for example, it's not possible to use inline styles in the HTML content in the editor. However, it is possible to use CSS class names, and then define the class in the `mail.css` file, which is then included in the emails. Therefore, it's recommended that you define a standard set of classes that the Developer Portal administrators can use when they're creating emails.

4. Optional: Add images to the `ibm-apic-mail-message.html.twig` template.  
Images can be used by adding a `` reference in the twig template. The `src` attribute for the image must be a fully qualified web URL, and must be externally accessible so that the email recipients can access the image. It's not possible to reference local images, they must be fully qualified URLs. It's also not possible to embed or attach images or other files in the emails.

Note: The `logo` will be automatically set to the fully qualified URL for the site logo image.

5. Compress all of your sub-theme resources back into a .zip file, and install into the Developer Portal. See [Installing additional themes](#) for details.

## What to do next

---

It is possible to have multiple email templates. For example for different modules, where one template could be used for the emails that are sent by the contact module, and another template for the emails that are sent by the Email subscribers wizard. This more advanced configuration can be done by using the module and key attributes, and is explained within the twig template itself. Essentially, you create a twig template that's called `ibm-apic-mail-message.contact.html.twig`, for example, to override the default template for only the contact module. Then the most specific template for a module is used by the Developer Portal, before falling back on the more general template.

## Related tasks

---

- [Creating a sub-theme](#)
- [Installing additional themes](#)
- [Applying a modified content type template](#)

## Working with roles in the Developer Portal

---

As a site administrator, you can create and customize site configuration roles in the Developer Portal.

Rather than assigning individual permissions to each user, permissions are assigned to roles, and then roles are assigned to users. Using roles gives administrators greater control over permissions, and makes it easy to assign and remove roles when necessary.

Administrators can assign the following site configuration roles to users in the Developer Portal:

### Superuser

The Superuser role has the highest level of permissions in the Developer Portal, and is the role that is assigned by default to the admin user, in addition to the Administrator role, when they enable a Developer Portal site in the API Manager. This role bypasses all content access control, which means that users that have a Superuser role can see all of the site content. This level of access can lead to some unexpected behavior when they access content such as their Consumer organizations and their applications. Careful consideration should be given before you assign users to a Superuser role.

### Administrator

An Administrator can use the administrator dashboard to customize and configure the Developer Portal. An Administrator is able to perform any task within the Developer Portal that does not involve the creation of APIs, Products, and Apps.

#### Content Author

A Content Author is a curator of content for the Developer Portal. A Content Author can perform the following actions:

- Set taxonomies on the content.
- Provide custom icons for content.
- Upload additional documentation files.
- Create documentation pages in the Developer Portal.

#### Forum Moderator

A Forum Moderator can moderate the content of a forum in the Developer Portal.

Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: **A user already exists with this email address.**

To learn how to create and customize roles in the Developer Portal, use the following topic links:

- [Creating administrator users for the Developer Portal](#)  
You can create additional administrator users for an Developer Portal.
- [Assigning users to a role](#)  
Assign users to one or more roles in the Developer Portal to enable role specific permissions.
- [Assigning permissions to a role](#)  
You can assign permissions to a new role in the Developer Portal, and update the permissions for current roles.
- [Creating a new role](#)  
You can create a new role within the Developer Portal.

---

## Creating administrator users for the Developer Portal

You can create additional administrator users for an Developer Portal.

---

### Before you begin

You must have administrator access to complete this task.

---

### About this task

After you enable a Developer Portal site in the API Manager, you are sent an email that contains a one-time log in link for the Developer Portal site Admin user. The Admin user can administer the Content Management System (CMS) capabilities of the Developer Portal. For more information about enabling a site, see [Creating and configuring Catalogs](#).

However, you can also create additional administrative roles for users from within the Developer Portal by assigning the role to users that have access to the Developer Portal. A user with an Administrator role can use the administrator dashboard to customize and configure the Developer Portal. An Administrator is able to perform any task within the Developer Portal that does not involve the creation of APIs, Products, and Apps. You should be careful to ensure that only trusted users are given this access and level of control of your site.

---

### Procedure

To create additional administrative users, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click People in the administrator dashboard.
3. Select the List tab.
4. Select the check boxes for the target users.
5. From the drop-down list under the Action heading, select Add the Administrator role to the selected user(s).
6. Click Apply to selected items.

---

### Results

You have assigned the selected users to the Administrator role.

---

## Assigning users to a role

Assign users to one or more roles in the Developer Portal to enable role specific permissions.

---

### Before you begin

You must have administrator access to complete this task.

## Procedure

---

To assign users to a role, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click People in the administrator dashboard.
3. Select the List tab.
4. Select the check boxes for the target users.
5. From the drop-down list under the Action heading, select the role that you want to assign.
6. Click Apply to selected items.

## Results

---

You have assigned the selected users to the required role.

---

## Assigning permissions to a role

You can assign permissions to a new role in the Developer Portal, and update the permissions for current roles.

## Before you begin

---

You must have administrator access to complete this task.

## Procedure

---

To assign or update permissions for a role, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click People in the administrator dashboard.
3. Select the Permissions tab.
4. Assign a permission to a role, by selecting the check box in the relevant column for the permission that you want to assign.
5. Click Save permissions.

## Results

---

You have assigned permissions to the target role.

---

## Creating a new role

You can create a new role within the Developer Portal.

## Before you begin

---

You must have administrator access to complete this task.

## Procedure

---

To create a new role, complete the following steps:

1. Click People in the administrator dashboard.
2. Select the Roles tab.
3. Click Add role.
4. Enter a name for the new role in Role name, and click Save.
5. Your new role is displayed in the list of roles. You can drag any of the roles to re-order them. It's recommended to order roles from least permissive (for example, Anonymous user) to most permissive (for example, Administrator user). Users who aren't logged in have the Anonymous user role. Users who are logged in have the Authenticated user role, plus any other roles granted to their user account.

## Results

---

You have created a new role in the Developer Portal.

## What to do next

---

The new role is created with the same permissions as an Authenticated user role. You can click the Permissions tab and edit the permissions as required.

---

## Forums

Create and control forums in the Developer Portal.

The Developer Portal lets you create threaded discussion boards, called forums, on your site. To learn how to configure forums, use the following topic links.

- [Creating a new forum](#)  
You can create a new forum in the Developer Portal.
- [Creating new forum containers](#)  
You can create new forum containers in the Developer Portal.
- [Configuring the creation of new forums for each API](#)  
You can choose whether or not to automatically create a new forum when an API is created in the Developer Portal.
- [Locking forum topics](#)  
You can lock forum topics in the Developer Portal.
- [Marking content in a forum as sticky](#)  
You can mark content in a forum as sticky in the Developer Portal.
- [Removing forum posts](#)  
You can remove forum posts in the Developer Portal.
- [Turning off forums in the Developer Portal](#)  
You can turn off forums in the Developer Portal by disabling the Forum module. This action prevents users from viewing, and contributing to, forums.

---

## Creating a new forum

You can create a new forum in the Developer Portal.

### Before you begin

---

You must have administrator or Forum moderator access to complete this task.

### Procedure

---

To create a new forum, complete the following tasks:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Forums.
4. Click Add forum.
5. Fill in the Name and Description fields for the new forum.
6. From the Parent drop-down list, select the hierarchical position of the new forum.
7. Click Save. You have created a new forum.

---

## Creating new forum containers

You can create new forum containers in the Developer Portal.

### Before you begin

---

You must have administrator or Forum moderator access to complete this task.

### Procedure

---

To create a new forum containers, complete the following tasks:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Forums.
4. Click Add container.
5. Fill in the Name and Description fields for the new forum container.
6. From the Parent drop-down list, select the hierarchical position of the new forum container.
7. Click Save. You have created a new forum container.

---

## Configuring the creation of new forums for each API

You can choose whether or not to automatically create a new forum when an API is created in the Developer Portal.

### Before you begin

---

You must have administrator access to complete this task.

### Procedure

---

To configure the creation of new forums for each API, complete the following tasks:

1. If the administrator dashboard is not displayed, click **Manage** to display it.
  2. Click **Configuration** in the administrator dashboard.
  3. Under the **SYSTEM** heading, click **IBM API Connect**.
  4. Select or clear the **Automatically create a forum per API** check box to turn the feature on or off.
  5. Click **Save configuration**.
- You have configured the creation of new forums for each API.

---

## Locking forum topics

You can lock forum topics in the Developer Portal.

### Before you begin

---

You must have administrator or Forum moderator access to complete this task.

### Procedure

---

To lock forum topics, complete the following tasks:

1. If the administrator dashboard is not displayed, click **Manage** to display it.
  2. Click **Content** in the administrator dashboard.
  3. Click **Edit** for the forum topic that you want to lock.
  4. Click **Comment settings**. The view expands.
  5. Select **Closed**.
  6. Keeping the **Published** selected, click **Save**.
- You have disallowed any further replies and marked the topic as locked.

---

## Marking content in a forum as sticky

You can mark content in a forum as sticky in the Developer Portal.

### Before you begin

---

You must have administrator or Forum moderator access to complete this task.

### Procedure

---

To mark content in a forum as sticky, complete the following tasks:

1. If the administrator dashboard is not displayed, click **Manage** to display it.
  2. Click **Content** in the administrator dashboard.
  3. From the **Action** drop-down menu, select **Make content sticky**.
  4. Select the check boxes next to the forum topics that you want to make sticky.
  5. Click **Apply** to selected items.
- You made the selected forum content sticky.

---

## Removing forum posts

You can remove forum posts in the Developer Portal.

### Before you begin

---

You must have administrator or Forum moderator access to complete this task.

### Procedure

---

To remove forum posts, complete the following tasks:

1. If the administrator dashboard is not displayed, click **Manage** to display it.
  2. Click **Content** in the administrator dashboard.
  3. From the **Action** drop-down menu, select **Delete content**.
  4. Select the check boxes next to the forum posts that you want to remove.
  5. Click **Apply** to selected items.
- You removed the selected forum posts.

---

# Turning off forums in the Developer Portal

You can turn off forums in the Developer Portal by disabling the Forum module. This action prevents users from viewing, and contributing to, forums.

---

## Before you begin

You must have administrator access to complete this task.

---

## About this task

To turn off forums, you must first remove all Forums terms, and then you can disable the Forum module.

---

## Procedure

To turn off forums, complete the following steps.

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Delete all Forums terms by completing the following steps:
  - a. Click Structure > Taxonomy > Forums in the administrator dashboard.  
The list of Forums terms is displayed.
  - b. For each item in the list of terms, select Delete from the drop-down list in the OPERATIONS column, and then click Delete again to confirm.  
All of the Forums terms are deleted.
3. Disable the Forum module by completing the following steps:
  - a. Click Extend > Disable module in the administrator dashboard.
  - b. Select the Forum module, then click Disable.
  - c. The display then shows a list of forum configuration that will also be removed if the Forum module is disabled. Click Disable again to confirm that you agree to this removal.

---

## Results

You successfully turned off forums in the Developer Portal.

If you want, you can enable the Forum module again by clicking Extend, selecting the Forum module on the List tab, and clicking Enable.

---

## Reports

You can view reports about the general configuration and status of your Developer Portal site.

Use the Reports section in the Developer Portal to view information about the following areas:

### Available updates

Status report about available updates for your installed modules and themes.

### Node Type Count

Status report about how much of each content type there is in your portal site, and whether it is published or not. It can also provide totals of how many users there are of each role.

### Available translation updates

Status report about available interface translations for your installed modules and themes.

### Field list

Overview of fields on all entity types.

### Status report

Short overview of your site's parameters, as well as any problems detected with your installation.

### Views plugins

Overview of the plugins that are used in all views.

---

## Extend

You can extend the Developer Portal by creating and installing custom modules, or installing additional Drupal contributed modules.

Modules provide functionality to your site, and the *core modules* that are included in the standard Developer Portal installation, provide all of the basic functions that most sites need. However, some of the core modules are not enabled by default, so you might want to examine the module list under the Extend menu of your site, to see whether the functionality you require is available from a core module before starting to investigate custom and contributed modules. If you still require additional custom or contributed modules, consider the following topics to help you create and implement modules consistently.

Note: The Developer Portal contains a Rules framework for event based actions. Some of those actions and triggers might exhibit unexpected behavior. A workaround is to create a custom module instead, because the same events and actions are possible there.

- [Custom module development: an introduction to Drupal tools for PHP development](#)  
Some guidance about the Drupal tools and environment considerations when developing in PHP, to help when writing custom modules to extend your Developer Portal functionality.
- [Custom module development: background and prerequisites](#)  
You can create custom modules in the Developer Portal to extend functionality. The following information provides a getting started developer guide to custom module development.

- [Custom module development: creating a module skeleton](#)  
You can extend your Developer Portal site by creating custom modules. The starting point for a new module is to create a module skeleton.
- [Custom module development: using hooks](#)  
You can use hooks in custom modules to alter the behavior of Drupal core or other modules in your Developer Portal.
- [Installing custom modules](#)  
You can extend your Developer Portal site by installing custom modules that you created, and also installing contributed modules from the Drupal community.
- [Deleting custom modules](#)  
You can delete custom modules from your Developer Portal if you want to remove them entirely from your site.
- [Disabling modules](#)  
You can disable an entire module in the Developer Portal if you want to improve performance, or remove functionality.
- [Using Developer Portal metadata on API Manager](#)  
For content created or updated after Version 10.0.1.0, the Developer Portal propagates more metadata to API Manager for `consumerorgs`, `users`, and `applications`.
- [Using a custom module for payment method creation](#)  
You can offer free use of your products to your customers or you can configure your Developer Portal to bill your customers when they use your products. You can use a custom module to configure creation of payment methods.

---

## Related information

- [📄 Extending Drupal](#)

---

## Custom module development: an introduction to Drupal tools for PHP development

Some guidance about the Drupal tools and environment considerations when developing in PHP, to help when writing custom modules to extend your Developer Portal functionality.

The following sections provide an overview of the key automation tools and environment considerations for PHP development in the Developer Portal. For a complete list of Drupal tools, see [Development tools overview](#) on Drupal.org, however, note that you are not permitted to access the shell inside the Developer Portal containers. Important:

- You are not permitted to include any IBM® API Connect modules within any custom modules that you create. Also, directly editing any API Connect themes, modules, included modules, or Drupal core on the file system is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed.
- All custom development is your responsibility. Although the use of custom modules and themes is supported, IBM API Connect do not provide support in their development or modification.

---

## Devel, Devel Generate Kint, and Webprofiler development modules

The Devel, Devel Generate Kint, and Webprofiler modules are all packaged within the Devel module. These modules provide the following development support:

- *Devel* - Helper functions for Drupal developers.
- *Devel Generate Kint* - Accelerate development of your site or module by quickly generating nodes, comments, terms, users, and more.
- *Kint* - Pretty print of variables with `kint($my_var)`.
- *Webprofiler* - Port of the Symfony WebProfiler Bundle as a Drupal module.

For more information, see [Devel module](#) on drupal.org, and [Devel](#) on drupalship.org.

---

## Admin Toolbar module

The Admin Toolbar module improves the default Drupal Toolbar by transforming it into a drop-down menu, providing fast access to the administration pages. It also adds extra links to the Drupal icon on the admin toolbar to development tools such as flush all caches, run cron, and run updates.

For more information, see [Admin toolbar module](#) on drupal.org.

---

## Examples for developers project

The Examples for Developers project aims to provide high-quality, well-documented API examples for a broad range of Drupal core functionality.

The project contains many modules that illustrate best practices for implementing Drupal core APIs. For example there are examples for Block, Cache, Config and Content Entity, Cron, Database API, Email, Events, Form API, Field, Field Permission, File, Hooks, Javascript, Node Type, Page, Pager, PHPUnit, Plugin Type, Queue, Simple Test, Stream Wrapper, Table Sort, Testing, and Tour.

For more information, see [Examples for developers project](#) on drupal.org.

---

## Coding standards

The Drupal Coding Standards apply to code within Drupal and its contributed modules. For more information, see [Coding standards](#) on drupal.org.

---

## Related tasks

- [Installing custom modules](#)

---

## Related information



- [📖 Drupal tools for developers](#)

---

## Custom module development: background and prerequisites

You can create custom modules in the Developer Portal to extend functionality. The following information provides a getting started developer guide to custom module development.

Custom modules are written in PHP, which is an open source server-side scripting language that is used for creating dynamic web pages. For general information on the Drupal tools for PHP development, see [Custom module development: an introduction to Drupal tools for PHP development](#).

The following sections provide an overview of custom module development, with links to learn more about the required development languages and concepts. Creating custom modules affects the source of your Developer Portal site, so it is recommended that you gain Drupal experience and PHP knowledge before you start creating modules.

Important:

- You are not permitted to include any IBM® API Connect modules within any custom modules that you create. Also, directly editing any API Connect themes, modules, included modules, or Drupal core on the file system is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed.
- All custom development is your responsibility. Although the use of custom modules and themes is supported, IBM API Connect do not provide support in their development or modification.

---

## Object-Oriented Programming (OOP)

OOP is established as best practice for development, so ensure your knowledge of OOP is good. The PHP documentation on [Classes and Objects](#) is a good place to start. For a general overview of PHP best practices, see [PHP The Right Way](#). The following resources are also helpful:

- [Object-oriented programming](#) (on Wikipedia)
- [Object Oriented Programming with PHP](#) (on phpro.org)
- [Object-Oriented PHP for Beginners](#) (on tutstplus.com)
- [Object Oriented Programming in PHP](#) (on tutorialspoint.com)

Drupal also uses some common design patterns, so you need to ensure you have a basic understanding of these. See the following pattern resources:

- [The Factory Pattern](#) (on phpthtrightway.com), and [Late Static Bindings](#) (on php.net)
- [Software design pattern](#) (on Wikipedia)
- [Programming Foundations: Design Patterns](#) (on lynda.com)

---

## PHP namespaces

You must be familiar with the concept of namespaces in PHP. In most cases, the Drupal code is namespaced based on the module that code belongs to.

For example, the namespace for `block.module` is:

```
namespace Drupal\block;
```

When you create Drupal modules, it is important to consider that PHP has a global namespace, and that function names must be unique. You are recommended to prefix the name of any methods with the module name.

For example, if you have a module that is named `custom_module`, a `create` method within it is called `custom_module_create()`.

For more information about namespaces, see the following resources:

- [Drupal namespace standards](#) (on drupal.org)
- [How to use PHP namespaces](#) (on sitepoint.com)
- [PHP Namespaces](#) (on php.net)

---

## Dependency injection

It is important that you have a baseline understanding of dependency injection. Drupal makes heavy use of this concept, so you will need this understanding to be able to access and make use of many of the core APIs.

To find out more about dependency injection, see [Dependency Injection](#) (on phpthtrightway.com), as well as the additional articles that are linked to on that page. See also [Services and dependency injection in Drupal](#) (on drupal.org).

---

## Symfony

Symfony is a set of reusable PHP components and a PHP framework for web projects. Drupal borrows from this framework in order to reduce code duplication across various PHP projects. Much of the code that Drupal uses to handle routing, sessions, and the services container, amongst other things, is borrowed from Symfony.

To understand how Symfony works, see the [Symfony Documentation](#) on symfony.com.

---

## Other useful resources

- [Drupal API Annotations](#) - list of the different annotation types that Drupal uses for plugin discovery, and to provide additional context/meta-data for the code that's being executed.
- [Plugin API overview](#) - overview about how plugins are used in Drupal.
- Tools are available for checking your custom modules, such as [drupal\\_check](#) on GitHub, which checks Drupal code for deprecations.

## Related tasks

---

- [Installing custom modules](#)

## Related information

---

- [Creating custom modules](#)

# Custom module development: creating a module skeleton

You can extend your Developer Portal site by creating custom modules. The starting point for a new module is to create a module skeleton.

## Before you begin

---

You must have administrator access to complete this task.

You must also have experience in Drupal and PHP development. For more information, see [Custom module development: an introduction to Drupal tools for PHP development](#) and [Custom module development: background and prerequisites](#).

Important:

- You are not permitted to include any IBM® API Connect modules within any custom modules that you create. Also, directly editing any API Connect themes, modules, included modules, or Drupal core on the file system is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed.
- All custom development is your responsibility. Although the use of custom modules and themes is supported, IBM API Connect do not provide support in their development or modification.

## About this task

---

Every custom module creation starts with creating an .info.yml file. This file, or module skeleton, is used to store the metadata about the custom theme. After the module skeleton is created, the module can be installed onto your site by using the Extend menu on the administrator dashboard of the Developer Portal UI. However, without any module specific code, your site won't be affected.

The following instructions guide you through creating a custom module skeleton. What to do next in your custom module creation depends on what you want your module to do. Options are shown at the end of this task.

## Procedure

---

1. Choose a short name, also known as a machine name, for your custom module.

The machine name is used in several files and functions in your module, as well as by Drupal core to programmatically refer to your module. The name must conform to the following rules:

- It must start with a letter.
- It must contain only lowercase letters and underscores. (Note that if uppercase letters are used in the name, Drupal won't recognize any hook implementations in your module.)
- It must not contain any spaces.
- It must be unique within your Developer Portal site.
- It should not be any of the following reserved terms: **src, lib, vendor, assets, css, files, images, js, misc, templates, includes, fixtures, Drupal.**

2. Create a module folder in a location of your choice, for example, `/modules/module_name`.

You don't have to name the folder the same name as your module machine name, however, you must remember to use the machine name programmatically within your module's code and filenames.

3. Create a `module_name.info.yml` file in your module folder.

Where `module_name` is the short name, or machine name, that you chose in Step 1.

4. Insert the metadata for your custom module into the `module_name.info.yml` file as follows:

```
name: Example module name
description: Description of the module.
package: Custom
```

```
type: module
core: 8.x
core_version_requirement: '^8 || ^9'
version: 1.0
```

```
dependencies:
 - module_name
```

```
test_dependencies:
 - module_name
```

```
configure: example.settings
```

```
php: 7.1
```

```
hidden: true
```

where

- **name** (required) is the name of your module, and is the text that is shown on the module administration page in the Developer Portal UI.
- **description** (optional) is a description of your module, and is the text that is shown on the module administration page in the Developer Portal UI.

- **package** (optional) is a key that enables you to group similar modules together.
- **type** (required) is the type of extension that you are creating, for example, a module, theme or profile. In this instance you are creating a module.
- **core** (optional) specifies with which Drupal core version your module is compatible.
- **core\_version\_requirement** (required) specifies which versions of Drupal your module is compatible with. This example specifies that the module is compatible with all versions of Drupal 8 and 9.
- **version** (required) is the version number of your module. Note that **version** is not required if you are publishing your module on Drupal.org.
- **dependencies** (optional) is a YAML list of any modules that your module depends on. They must be namespaced in the format `module_name`, where `module_name` is the module's machine name.
- **test\_dependencies** (optional) is list of any modules (in the same format as **dependencies**) that are needed to run certain automated tests for your module on Drupal's automated test runner (DrupalCI), but are not needed as module dependencies in general.
- **configure** (optional), if your module offers a configuration form, specify the route to this form here.
- **php** (optional) is the minimum PHP version that is required for your module. For the Developer Portal the minimum PHP version is 7.1.
- **hidden** (optional) is an option that, if set, will hide your module from the Extend section of the administrator dashboard on the Developer Portal UI. This option might be useful if your module only contains tests, or if the module is meant as an example to other developers.

5. Save your `module_name.info.yml`.

## Results

You successfully created your module skeleton.

## Example

An example `.info.yml` file:

```
name: User field example
type: module
description: Developer Portal tutorial about a user field example
package: IBM API Connect Tutorials
core: 8.x
core_version_requirement: '^8 || ^9'
version: 8.x-0.0.1
dependencies:
 - ibm_apim
 - auth_apic
```

## What to do next

You now need to create the appropriate content for your custom module, depending on what you want the module to do. For information about the various options, see [Creating custom modules](#) on Drupal.org. There is also a walkthrough example of creating a basic custom module that is available on Drupal.org, see: [A "Hello World" custom page module](#).

Note: If you are defining URL paths in the routing file of your module, you must consider any potential implications of those paths, and avoid conflicts with other parts of the Developer Portal. For example, you are recommended not to prefix any paths with `ibm_apic`.

After you have created your custom module content, you need to package your custom module files up into a zip, tar, `tgz`, `gz`, or `bz2` file. This file can then be installed onto your Developer Portal site by using the Extend menu on the administrator dashboard. For more information, see [Installing custom modules](#).

## Related information

- [Let Drupal know about your module with an `.info.yml` file](#)

## Custom module development: using hooks

You can use hooks in custom modules to alter the behavior of Drupal core or other modules in your Developer Portal.

Custom modules enable you to extend the functionality of your Developer Portal site. For more information about how to create custom modules, see [Custom module development: an introduction to Drupal tools for PHP development](#), and [Custom module development: background and prerequisites](#).

Hooks are one of the ways that custom modules can use to interact with other modules and with Drupal core subsystems. The following sections provide an overview of hooks, and a list of the IBM® API Connect specific hooks.

Important:

- You are not permitted to include any IBM API Connect modules within any custom modules that you create. Also, directly editing any API Connect themes, modules, included modules, or Drupal core on the file system is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed.
- All custom development is your responsibility. Although the use of custom modules and themes is supported, IBM API Connect do not provide support in their development or modification.

## About hooks

Hooks define functions that alter the behavior of Drupal core. So one way for custom modules to alter these functions, is to use hooks. Hooks are specially-named functions that a module defines (this is known as *implementing the hook*), these hooks are then discovered and called at specific times to alter or add to the base behavior or data (this is known as *invoking the hook*). Each hook has a name (for example: `hook_batch_alter()`), a defined set of parameters, and a defined return value. Your custom modules can implement hooks that are defined by Drupal core, by API Connect, or by other modules that they interact with. Your custom modules can also define their own hooks in order to let other modules interact with them. For more information, see [Understanding Hooks](#) on Drupal.org.

Note: If you write a hook implementation function, the function must not throw errors or exceptions, or use `alert` level log messages. Errors, exceptions, and `alert` level log messages can break all the processing in the parent code. For example, if you write a hook that calls an external server when an application is updated, the code needs

to have error handling in place to handle the server not being found, or not returning the expected result. Otherwise, Drupal stops parsing the function and the application might not get updated correctly. It can cause webhook and snapshot parsing to abort leaving the portal in an inconsistent state. For a list of all the hooks that are available on Drupal core, and how to implement them in your custom module, see [Drupal API Hooks](#).

For a list of all the API Connect specific hooks, see the following section.

## Using hooks in custom modules

To invoke a hook you need to add a function to your .module file in your custom module, and prefix the hook with your custom module name. For example, when using an API Connect hook in a custom module, you must replace the word `hook` in the hook name with the name of your custom module. For example:

`hook_apic_app_create`

should be referenced in your custom module as:

`moduleName_apic_app_create`

See also the following example .module file that shows a custom module called `user_field_example`:

```
/**
 * Implementation of hook_form_alter() to alter the Sign up form
 */
function user_field_example_form_alter(&$form, FormStateInterface $form_state, $form_id) {
 if ($form_id === 'user_register_form') {
 // add our validator to the #validate array for the user_register_form
 $form['#validate'][] = 'user_field_example_validate_department_code';
 }
}

/**
 * Validate the Department code field on the Sign up form.
 *
 * Valid entry = DEPnnn
 * where n = single figure digit.
 */
function user_field_example_validate_department_code($form, &$form_state) {
 if (isset($form_state->getValue('field_department_code')[0]['value'])) {
 $dept_code = $form_state->getValue('field_department_code')[0]['value'];
 $valid = preg_match('/^DEP\d{3}$/i', $dept_code);
 if (!$valid) {
 // if the value is not valid then set an inline error on the relevant field
 $form_state->setErrorByName('field_department_code', t('Invalid department code.'));
 }
 }
}
```

## API Connect specific hooks

The following tables list the hooks that are specific to the API Connect Developer Portal. Each section contains a link to the .php file on the API Connect Developer Portal repository on GitHub for those hooks. This file contains examples of how you can use the hooks in your modules.

Hooks about Applications

The following table lists the hooks that relate to Applications in the Developer Portal. For examples of how to use Application hooks, see [apic\\_app/apic\\_app.api.php](#) on GitHub.

Table 1. Hooks about Applications

Hook name	Description
<code>hook_apic_app_create</code>	Triggered when an Application is created.
<code>hook_apic_app_update</code>	Triggered when an Application is updated.
<code>hook_apic_app_pre_delete</code>	Triggered when an Application is deleted, before the node deletion or cascade has happened.
<code>hook_apic_app_post_delete</code>	Triggered when an Application is deleted, after the node deletion or cascade has happened.
<code>hook_apic_app_promote</code>	Triggered when an Application is promoted.
<code>hook_apic_app_creds_create</code>	Triggered when a new set of credentials is created for an Application.
<code>hook_apic_app_creds_update</code>	Triggered when a set of credentials is updated for an Application.
<code>hook_apic_app_creds_delete</code>	Triggered when a set of credentials is deleted for an Application.
<code>hook_apic_app_subscribe</code>	Triggered when a subscription is created.
<code>hook_apic_app_migrate</code>	Triggered when a subscription is migrated to a new Plan.
<code>hook_apic_app_unsubscribe</code>	Triggered when an Application is unsubscribed from a Plan.
<code>hook_apic_app_image_create</code>	Triggered when a custom Application image is created.
<code>hook_apic_app_image_delete</code>	Triggered when a custom Application image is deleted.
<code>hook_apic_app_clientid_reset</code>	Triggered when a credential client ID is reset.
<code>hook_apic_app_clientsecret_reset</code>	Triggered when a credential client secret is reset.
<code>hook_apic_app_modify_getplaceholderimage_alter</code>	Alter the Application placeholder image provided in <code>\Drupal\apic_app\Application::getPlaceholderImage()</code> . Can be used to define a specific placeholder image to use when the consumer has not uploaded their own custom image for their Application.
<code>hook_apic_app_modify_getimageforapp_alter</code>	Alter the Application placeholder image provided in <code>\Drupal\apic_app\Application::getImageForApp()</code> . Can be used to provide a full path to a specific image to use for an Application that overrides any custom image that might have been uploaded.

Hook name	Description
<code>hook_apic_app_modify_client_id_reset_alter</code>	Alter the client ID provided by API Manager when the ID is reset.
<code>hook_apic_app_modify_client_secret_reset_alter</code>	Alter the client secret provided by API Manager when the secret is reset.
<code>hook_apic_app_modify_credentials_alter</code>	Alter the credentials provided by API Manager when a new Application is created.
<code>hook_apic_app_modify_credentials_create_alter</code>	Alter the credentials provided by API Manager when new credentials are created.

Hooks about Consumer organizations

The following table lists the hooks that relate to Consumer organizations in the Developer Portal. For examples of how to use Consumer organization hooks, see [consumerorg/consumerorg.api.php](https://github.com/consumerorg/consumerorg.api.php) on GitHub.

Table 2. Hooks about Consumer organizations

Hook name	Description
<code>hook_consumerorg_create</code>	Triggered when a Consumer organization is created.
<code>hook_consumerorg_update</code>	Triggered when a Consumer organization is updated.
<code>hook_consumerorg_pre_delete</code>	Triggered when a Consumer organization is deleted, before the node deletion or cascade has happened.
<code>hook_consumerorg_post_delete</code>	Triggered when a Consumer organization is deleted, after the node deletion or cascade has happened.
<code>hook_consumerorg_payment_method_create_alter</code>	Triggered to allow modification of the payment method creation form.
<code>hook_consumerorg_myorg_tabs_alter</code>	Triggered to allow more tabs to be added to the <code>my organization</code> page.

Hooks about APIs

The following table lists the hooks that relate to APIs in the Developer Portal. For examples of how to use API hooks, see [apic\\_api/apic\\_api.api.php](https://github.com/apic_api/apic_api.api.php) on GitHub.

Table 3. Hooks about APIs

Hook name	Description
<code>hook_apic_api_create</code>	Triggered when an API is created.
<code>hook_apic_api_update</code>	Triggered when an API is updated.
<code>hook_apic_api_delete</code>	Triggered when an API is deleted.

Hooks about Products

The following table lists the hooks that relate to Products in the Developer Portal. For examples of how to use Product hooks, see [product/product.api.php](https://github.com/product/product.api.php) on GitHub.

Table 4. Hooks about Products

Hook name	Description
<code>hook_product_create</code>	Triggered when a Product is created.
<code>hook_product_update</code>	Triggered when a Product is updated.
<code>hook_product_delete</code>	Triggered when a Product is deleted.

Other IBM API Connect Hooks

The following table lists the hooks that can be for other uses.

Table 5. Other APIC Hooks

Hook name	Description
<code>hook_ibm_apim_subscription_wizard_summary_alter</code>	Triggered to allow modification of the summary page of the subscription wizard.

## Related tasks

- [Installing custom modules](#)

## Related information

- [Creating custom modules](#)

## Installing custom modules

You can extend your Developer Portal site by installing custom modules that you created, and also installing contributed modules from the Drupal community.

## Before you begin

You must have administrator access to complete this task.

## About this task

You can add extra functions to your site by installing new custom or contributed modules. For more information on creating custom modules, see [Custom module development: background and prerequisites](#). Contributed modules are modules that are contributed by Drupal community members. To search for contributed modules, go to [Drupal Download & Extend](#). You can install both types of modules into your Developer Portal site by using the following instructions.

Note: The following modules are unsupported and their installation is blocked in the Developer Portal:

- `backup_migrate`
- `content_sync`
- `delete_all`
- `devel_themer`
- `domain`

- `php`
- `simple_html_mail`
- `swiftmailer`
- `tfa`
- `theme_editor`

---

## Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. In the administrator dashboard, click Extend.  
The List tab for the Extend page opens, and the list of installed modules is displayed. The list shows all the modules that are installed. The modules that are displayed with a selected checkbox are enabled, the modules that are displayed without a selected checkbox are not enabled.
3. Click + Install new module, and either complete the Install from a URL field, or click Choose file to upload a *filename.tar.gz* module file from your local computer.
4. Click Install.  
The Update manager confirms that the module was installed successfully.
5. Click Enable newly added modules to return to the List tab for the Extend page.
6. Find your newly added module in the list of modules, select its checkbox, and click Enable.  
Your newly added module is now enabled and available for you to use in the Developer Portal.

---

## Deleting custom modules

You can delete custom modules from your Developer Portal if you want to remove them entirely from your site.

---

### Before you begin

You must have administrator access to complete this task.

Note that any custom modules that you want to delete must be disabled first, including disabling any associated sub-modules. For more information, see [Disabling modules](#). Any modules that are shipped with IBM® API Connect cannot be deleted.

---

## Procedure

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. In the administrator dashboard, click Extend.
4. Click the Delete tab.  
The list of disabled custom modules that can be deleted is displayed. Note that any disabled modules that have sub-modules that are still enabled, are prevented from being disabled.
5. After you have found the module that you want to disable, select the check box next to the module, then click Delete.
6. At the confirmation dialogue, select Delete to delete the module completely from your site, or select Cancel to cancel the operation.

---

## Results

You successfully deleted the module.

---

### What to do next

To install a module, see [Installing custom modules](#).

---

## Disabling modules

You can disable an entire module in the Developer Portal if you want to improve performance, or remove functionality.

---

### Before you begin

You must have administrator access to complete this task.

Note that the disable process removes all data related to a module.

---

## Procedure

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. In the administrator dashboard, click Extend.  
The List tab for the Extend page opens, and the list of installed modules is displayed. The list shows all the modules that are installed, some of which are enabled (displayed with a selected check box), and some of which are not enabled (where the check box is not selected).
4. Click the Disable tab.  
The list of modules that can be disabled is displayed. Note that any modules that are required by other enabled modules, are prevented from being disabled.
5. Either enter the name of the module that you want to disable in the Filter by name or description field, or scroll through the list of modules.
6. After you have identified the module that you want to disable, select the check box next to the module, then click Disable.

7. At the confirmation dialogue, select Disable to disable the module and remove any data that relates to it, or select Cancel to cancel the operation.

## Results

---

You successfully disabled the module.

## What to do next

---

To enable a module, return to the List tab, select the check box next to the module that you want to enable, and click Enable. You can also delete custom modules completely from your Developer Portal site; see [Deleting custom modules](#).

## Using Developer Portal metadata on API Manager

---

For content created or updated after Version 10.0.1.0, the Developer Portal propagates more metadata to API Manager for **consumerorgs**, **users**, and **applications**.

This metadata might be values that the user provided by adding extra fields to content types, or added programmatically by using custom modules.

The benefit of this is that when you add custom fields to **consumerorgs**, **users**, and **applications**, this data is now also stored as metadata in API Manager. Storing this data allows the Developer Portal to correct or repopulate the custom fields from a webhook or a snapshot.

## Related tasks

---

- [Installing custom modules](#)

## Related information

---

- [Creating custom modules](#)

## Using a custom module for payment method creation

---

You can offer free use of your products to your customers or you can configure your Developer Portal to bill your customers when they use your products. You can use a custom module to configure creation of payment methods.

## Before you begin

---

You must have administrator access to complete this task.

You must have configured billing support in API Manager.

Create your custom module. Your custom module needs to implement the `consumerorg_payment_method_create_alter` hook, so it needs to have a function like this:

```
my_custom_module_consumerorg_payment_method_create_alter(&$form, $integration, $billing) {
 // do this payment function with $form
}
```

For more information, see [Custom module development: background and prerequisites](#).

## Procedure

---

1. If the administrator dashboard is not displayed, click Manage to display it.
2. In the administrator dashboard, click Extend.  
The List tab for the Extend page opens, and the list of installed modules is displayed. The list shows all the modules that are installed. The enabled modules are displayed with a selected checkbox. The disabled modules are displayed without a selected checkbox.
3. Click + Install new module, and either complete the Install from a URL field, or click Choose file to upload a *filename.tar.gz* module file from your local computer.
4. Click Install.  
The Update manager confirms that the module was installed successfully.
5. Click Enable newly added modules to return to the List tab for the Extend page.
6. Find your newly added module in the list of modules, select its checkbox, and click Enable.  
Your newly added module is now enabled and available for you to use in the Developer Portal.
7. In the administrator dashboard, click Configuration > System > IBM API Connect Billing Settings in the administrator dashboard.
8. Change the drop-down option to be your newly enabled module for your billing provider.

## Results

---

You successfully enabled your payment method by using a custom module.

## Managing the Developer Portal by using the toolkit CLI

---

You can perform operations on your Developer Portal by using the API Connect toolkit CLI.

- [Getting started with the Portal CLI commands](#)

The toolkit CLI provides developer portal-specific commands for managing your portal servers and sites.

- [Developer Portal CLI commands](#)

A summary of the commands that are supported in the Developer Portal CLI.

- [Scenarios that use the Portal CLI](#)

Here are some scenarios of how you can use the Portal commands in the toolkit CLI to accomplish tasks that are more complex to complete, or not possible in the Developer Portal UI.

---

## Getting started with the Portal CLI commands

The toolkit CLI provides developer portal-specific commands for managing your portal servers and sites.

### Procedure

---

1. Install the toolkit: [Setting up the API Connect toolkit](#)

2. Log in to the Management server: [Toolkit CLI Login](#).

```
apic login --server <management hostname/api base url> --username <username> --realm <realm>
```

Where:

- *<management hostname/api base url>* is the FQDN or base URL of your management API endpoints, for example: `mgmt.example.com` or `https://mgmt.example.com`.
- *<username>* is either the admin user or provider org owner, depending on whether you want to run cloud or provider organization scoped operations.
- *<realm>* is the user registry realm for the specified *<username>*. By default this is `admin/default-idp-1` for the admin user, and `provider/default-idp-2` for provider organization users.

Note: All portal CLI commands are either cloud admin or provider org scoped.

3. View the available toolkit portal commands by using `-m portaladmin -h`. Example:

```
apic -m portaladmin -h
...Usage:
 apic [flags]
 apic [command]
```

Available Flags:

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>-h, --help</code>	Help for apic
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

Available Commands:

Creating applications and artifacts	
config	Manage configuration variables
Publishing to the cloud	
api	Api operations
apic-config	Apic Config operations
application	Application operations
backups	Backups operations
consumer-org	Consumer Org operations
custom-module	Custom Module operations
custom-theme	Custom Theme operations
custom-translation	Custom Translation operations
custom-webserver-page	Custom Webserver Page operations
drupal-config	Drupal Config operations
drupal-state	Drupal State operations
entity	Entity operations
factory-reset	Factory Reset operations
ip-security-enabled	Ip Security Enabled operations
modules	Modules operations
php-memory	Php Memory operations
platforms	Platforms operations
product	Product operations
security	Security operations
service-ip-allowlist	Service Ip Allowlist operations
site	Site operations
site-config	Site Config operations
sites	Sites operations
themes	Themes operations
twig	Twig operations
Other commands	
licenses	Review the license for API Connect
validate	Validate an API or product definition
version	Get the APICConnect toolkit version

Additional help topics:

Use `"apic [command] --help"` for more information about a command.

4. To view command-specific help, run the command with `-h`:



```

apic -m portaladmin php-memory -h
Usage: apic php-memory:list [flags]

Php Memory operations

Flags:
 --format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
 -h, --help Help for list
 --portal_service_name string The name of the portal service (required)
 -s, --server string management server endpoint (required)

Type "apic [command] --help" for help on the following related commands:
 php-memory:list List the PHP memory limit on the platform.
 php-memory:update Set the PHP memory limit on the portal platform.

apic -m portaladmin php-memory:list -h
Usage: apic php-memory:list [flags]

List the PHP memory limit on the platform.

Lists the PHP memory_limit setting on the portal platform.

Flags:
 --format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
 --portal_service_name string The name of the portal service (required)
 -s, --server string management server endpoint (required)

```

## Developer Portal CLI commands

A summary of the commands that are supported in the Developer Portal CLI.

### Authenticating

Use the `apic login` command to authenticate, and the `apic logout` command to remove your local authentication credentials. For more information, see [Getting started with the Portal CLI commands](#).

Note: When you authenticate successfully, your credentials are stored, in plain text, in the file `Linux .netrc` or `Windows _netrc`. Therefore, you must set the file permissions in such a way that your credentials are not accessible by others.

### Command summary

The following tables summarize `apic --mode portaladmin` commands.

Table 1. Summary of general-purpose commands

Command	Description	Subcommands
<code>apic api</code>	Manage your <code>api</code> .	<ul style="list-style-type: none"> <li><code>apic api:add-attachment</code> - Add an attachment for an API</li> <li><code>apic api:add-tag</code> - Add a tag for an API.</li> <li><code>apic api:get</code> - Get an API from a portal service.</li> <li><code>apic api:get-document</code> - Get a specific entire API document from the portal of the given organization and catalog.</li> <li><code>apic api:list</code> - List the APIs of the given organization and catalog.</li> <li><code>apic api:set-icon</code> - Set an icon for an API.</li> </ul> <p> <code>-s, --server mgmt_endpoint_url</code>. Use the value of the platform API endpoint URL for this entry.  <code>--org orgid</code>  <code>--catalog catalog id</code>.  <code>--debug</code>. If you need debug information. </p> <p>For more information, see <a href="#">Using the api commands</a>.</p>
<code>apic apic-config</code>	Manage your <code>apic-config</code> .	<ul style="list-style-type: none"> <li><code>apic apic-config:get</code> - Get the configuration of a site.</li> </ul> <p> <code>-s, --server mgmt_endpoint_url</code>. Use the value of the platform API endpoint URL for this entry.  <code>--org orgid</code>  <code>--catalog catalog id</code>.  <code>--debug</code>. If you need debug information. </p> <p>For more information, see <a href="#">Using the apic-config commands</a>.</p>
<code>apic application</code>	Manage your <code>application</code> .	<ul style="list-style-type: none"> <li><code>apic application:get</code> - Get an application from a portal service.</li> <li><code>apic application:list</code> - List the applications on a portal service.</li> </ul> <p> <code>-s, --server mgmt_endpoint_url</code>. Use the value of the platform API endpoint URL for this entry.  <code>--org orgid</code>  <code>--catalog catalog id</code>.  <code>--debug</code>. If you need debug information. </p> <p>For more information, see <a href="#">Using the application commands</a>.</p>

Command	Description	Subcommands
<code>apic backups</code>	Manage your <code>backups</code> .	<ul style="list-style-type: none"> <li>• <code>apic backups:list</code> - List the backups of a portal service.</li> <li>• <code>-s, --server management_server</code>.</li> <li>• <code>--portal_service_name portal_service_name</code></li> <li>• <code>--format format</code>. The options are <code>json</code> or <code>yaml</code>.</li> </ul> <p>For more information, see <a href="#">Using the backups commands</a>.</p>
<code>apic consumer-org</code>	Manage your <code>apic consumer-org</code> .	<ul style="list-style-type: none"> <li>• <code>apic consumer-org:get</code> - Get a consumer org from a portal service.</li> <li>• <code>apic consumer-org:list</code> - List the consumer orgs on a portal service.</li> <li>• <code>-s, --server mgmt_endpoint_url</code>. Use the value of the platform API endpoint URL for this entry.</li> <li>• <code>--org orgid</code></li> <li>• <code>--catalog catalog id</code>.</li> <li>• <code>--debug</code>. If you need debug information.</li> </ul> <p>For more information, see <a href="#">Using the consumer-org commands</a>.</p>
<code>apic content</code>	Manage your <code>content</code> .	<ul style="list-style-type: none"> <li>• <code>apic content:list</code> - List all of the entities in your site for the given content type and bundle.</li> <li>• <code>apic content:list-type</code> - List the allowed content types on your site.</li> <li>• <code>-s, --server mgmt_endpoint_url</code>. Use the value of the platform API endpoint URL for this entry.</li> <li>• <code>--org orgid</code></li> <li>• <code>--catalog catalog id</code>.</li> <li>• <code>--format format</code>. Note that this flag has no affect on the <code>create-export</code> and <code>create-import</code> sub commands.</li> <li>• <code>--debug</code>. If you need debug information.</li> </ul> <p>For more information, see <a href="#">Using the content commands</a>.</p>
<code>apic custom-module</code>	Manage your <code>custom modules</code> .	<ul style="list-style-type: none"> <li>• <code>apic custom-module:create-export</code> - Create an export of a <code>custom module</code>.</li> <li>• <code>apic custom-module:create-import</code> - Create an import of a <code>custom module</code>.</li> <li>• <code>apic custom-module:delete-export</code> - Delete an export of a <code>custom module</code>.</li> <li>• <code>apic custom-module:delete-import</code> - Delete an import of a <code>custom module</code>.</li> <li>• <code>apic custom-module:get-export</code> - Get an export of a <code>custom module</code>.</li> <li>• <code>apic custom-module:get-export-status</code> - Get the status of an export of a <code>custom module</code>.</li> <li>• <code>apic custom-module:get-import-status</code> - Get the status of an import of a <code>custom module</code>.</li> <li>• <code>-s, --server mgmt_endpoint_url</code>. Use the value of the platform API endpoint URL for this entry.</li> <li>• <code>--org orgid</code></li> <li>• <code>--catalog catalog id</code>.</li> <li>• <code>--format format</code>. Note that this flag has no affect on the <code>create-export</code> and <code>create-import</code> subcommands.</li> <li>• <code>--debug</code>. If you need debug information.</li> </ul> <p>For more information, see <a href="#">Using the custom-module commands</a>.</p>
<code>apic custom-theme</code>	Manage your <code>custom themes</code> .	<ul style="list-style-type: none"> <li>• <code>apic custom-theme:create-export</code> - Create an export of a <code>custom theme</code>.</li> <li>• <code>apic custom-theme:create-import</code> - Create an import of a <code>custom theme</code>.</li> <li>• <code>apic custom-theme:delete-export</code> - Delete an export of a <code>custom theme</code>.</li> <li>• <code>apic custom-theme:delete-import</code> - Delete an import of a <code>custom theme</code>.</li> <li>• <code>apic custom-theme:get-export</code> - Get an export of a <code>custom theme</code>.</li> <li>• <code>apic custom-theme:get-export-status</code> - Get the status of an export of a <code>custom theme</code>.</li> <li>• <code>apic custom-theme:get-import-status</code> - Get the status of an import of a <code>custom theme</code>.</li> <li>• <code>-s, --server mgmt_endpoint_url</code>. Use the value of the platform API endpoint URL for this entry.</li> <li>• <code>--org orgid</code></li> <li>• <code>--catalog catalog id</code>.</li> <li>• <code>--format format</code>. Note that this flag has no affect on the <code>create-export</code> and <code>create-import</code> subcommands.</li> <li>• <code>--debug</code>. If you need debug information.</li> </ul> <p>For more information, see <a href="#">Using the custom-theme commands</a>.</p>

Command	Description	Subcommands
<code>apic custom-translation</code>	Manage your <code>custom translations</code> .	<ul style="list-style-type: none"> <li>• <code>apic custom-translation:create-export</code> - Create a task to export a <code>custom translation</code>.</li> <li>• <code>apic custom-translation:create-import</code> - Create a task to import a <code>custom translation</code>.</li> <li>• <code>apic custom-translation:delete-export</code> - Cancel any ongoing or recently created export tasks.</li> <li>• <code>apic custom-translation:delete-import</code> - Cancel any ongoing or recently created import tasks.</li> <li>• <code>apic custom-translation:get-export</code> - Streams an export artifact of a <code>custom translation</code> back to the provided task ID.</li> <li>• <code>apic custom-translation:get-export-status</code> - Get the status of an export of a <code>custom translation</code>.</li> <li>• <code>apic custom-translation:get-import-status</code> - Get the status of an import of a <code>custom translation</code>.</li> </ul> <p>For more information, see <a href="#">Using the custom-translation commands</a>.</p>
<code>apic custom-webserver-page</code>	Manage your <code>custom-webserver-page</code> content.	<ul style="list-style-type: none"> <li>• <code>apic custom-webserver-page:delete</code> - Delete a custom HTML web server error page.</li> <li>• <code>apic custom-webserver-page:get</code> - Get the content of a custom HTML web server error page.</li> <li>• <code>apic custom-webserver-page:set</code> - Set a custom HTML web server error page by using a valid HTML5 content file.</li> </ul> <p>For more information, see <a href="#">Using the custom-webserver-page commands</a>.</p>
<code>apic drupal-config</code>	Manage your <code>drupal-config</code> .	<ul style="list-style-type: none"> <li>• <code>apic drupal-config:delete</code> - Delete a Drupal config object, or a specific config key.</li> <li>• <code>apic drupal-config:get</code> - Get all of the key values for a Drupal config object, or a specific config key value.</li> <li>• <code>apic drupal-config:list</code> - List the Drupal config names by prefix, or list all.</li> <li>• <code>apic drupal-config:set</code> - Set a new Drupal config key and value, or update an existing config key and value.</li> </ul> <p>For more information, see <a href="#">Using the drupal-config commands</a>.</p>
<code>apic drupal-state</code>	Manage your <code>drupal-state</code> keys and values.	<ul style="list-style-type: none"> <li>• <code>apic drupal-state:delete</code> - Delete a specific Drupal state key and value.</li> <li>• <code>apic drupal-state:get</code> - Get a specific Drupal state key value.</li> <li>• <code>apic drupal-state:set</code> - Set a new Drupal state key and value, or update an existing state key value.</li> </ul> <p>For more information, see <a href="#">Using the drupal-state commands</a>.</p>
<code>apic entity</code>	Manage your <code>entity</code> .	<ul style="list-style-type: none"> <li>• <code>apic entity:count</code> - A count of entities from a site.</li> <li>• <code>-s, --server mgmt_endpoint_url</code>. Use the value of the platform API endpoint URL for this entry.</li> <li>• <code>--org orgid</code></li> <li>• <code>--catalog catalog id</code>.</li> <li>• <code>--debug</code>. If you need debug information.</li> </ul> <p>For more information, see <a href="#">Using the entity commands</a>.</p>
<code>apic export-entity</code>	Manage the export of your entities.	
<code>apic factory-reset</code>	Perform a factory reset of your Developer Portal.	<ul style="list-style-type: none"> <li>• <code>apic factory-reset:delete</code> - Perform a factory reset of your Developer Portal, deleting the portal service and all of its associated Developer Portal sites.</li> <li>• <code>--execute_reset</code> must be set to <code>true</code> to trigger the Developer Portal reset. If not set, the command performs a dry run status of a factory reset.</li> <li>• <code>--portal_service_endpoint portal_service_endpoint</code>.</li> <li>• <code>-s, --server management_server</code>.</li> </ul> <p>Warning: The <code>factory-reset</code> command is irreversible. You must ensure that you have backups created before running this command.</p> <p>For more information, see <a href="#">Using the factory-reset command</a>.</p>
<code>apic forums</code>	Manage your <code>forums</code> .	<ul style="list-style-type: none"> <li>• <code>apic forums:disable</code> - disables the Forum module for a given site.</li> <li>• <code>apic forums:enable</code> - enables the Forum module for a given site.</li> <li>• <code>-s, --server mgmt_endpoint_url</code>.</li> <li>• <code>--org orgid</code></li> <li>• <code>--catalog catalog id</code>.</li> </ul> <p>For more information, see <a href="#">Using the forums commands</a>.</p>
<code>apic ip-security-enabled</code>	Manage your <code>IP security</code> .	<ul style="list-style-type: none"> <li>• <code>apic ip-security-enabled:update</code> - Enable or disable the IP security at the portal service level.</li> <li>• <code>--enabled</code> Set to <code>true</code> or <code>false</code> to enable or disable IP security.</li> <li>• <code>--portal_service_name portal_service_name</code>.</li> <li>• <code>-s, --server management_server</code>.</li> </ul> <p>For more information, see <a href="#">Using the ip-security-enabled command</a>.</p>

Command	Description	Subcommands
<code>apic modules</code>	Manage your <code>modules</code> .	<ul style="list-style-type: none"> <li><code>apic modules:delete</code> - Delete <code>modules</code>.</li> <li><code>apic modules:disable</code> - Disable <code>modules</code>.</li> <li><code>apic modules:enable</code> - Enable <code>modules</code>.</li> <li><code>apic modules:list</code> - Get a list of your <code>modules</code>.</li> </ul> <p>For more information, see <a href="#">Using the modules commands</a>.</p>
<code>apic php-memory</code>	Manage your <code>php-memory</code> .	<ul style="list-style-type: none"> <li><code>apic php-memory:list</code> - List the PHP memory limit on the platform.</li> <li><code>apic php-memory:update</code> - Set the PHP memory limit on the portal platform.</li> <li><code>-s, --server mgmt_endpoint_url</code>. Use the value of the platform API endpoint URL for this entry.</li> <li><code>--debug</code>. If you need debug information.</li> </ul> <p>For more information, see <a href="#">Using the php-memory commands</a>.</p>
<code>apic platforms</code>	Manage your <code>platforms</code> .	<ul style="list-style-type: none"> <li><code>apic platforms:list</code> - List the platforms of a portal service.</li> <li><code>-s, --server management_server</code>.</li> <li><code>--portal_service_name portal_service_name</code></li> <li><code>--format format</code>. The options are <code>json</code> or <code>yaml</code>.</li> </ul> <p>For more information, see <a href="#">Using the platforms commands</a>.</p>
<code>apic product</code>	Manage your <code>product</code> .	<ul style="list-style-type: none"> <li><code>apic product:add-attachment</code> - Add an attachment for the given product.</li> <li><code>apic product:add-tag</code> - Add a tag for the given product.</li> <li><code>apic product:get</code> - Get a product from a portal service.</li> <li><code>apic product:get-document</code> - Get a specific entire product document from the portal of the given organization and catalog.</li> <li><code>apic product:list</code> - List the products on a portal service.</li> <li><code>apic product:set-icon</code> - Set an icon image for the given product.</li> <li><code>-s, --server mgmt_endpoint_url</code>. Use the value of the platform API endpoint URL for this entry.</li> <li><code>--org orgid</code></li> <li><code>--catalog catalog id</code>.</li> <li><code>--debug</code>. If you need debug information.</li> </ul> <p>For more information, see <a href="#">Using the product commands</a>.</p>
<code>apic security</code>	Manage your <code>security</code> .	<ul style="list-style-type: none"> <li><code>apic security:clear-bans</code> - Clear all of the banned user/IP addresses on a portal site.</li> <li><code>-s, --server mgmt_endpoint_url</code>. Use the value of the platform API endpoint URL for this entry.</li> <li><code>--org orgid</code></li> <li><code>--catalog catalog id</code>.</li> </ul> <p>For more information, see <a href="#">Using the security command</a>.</p>
<code>apic service</code>	Manage your <code>service</code> operations.	<ul style="list-style-type: none"> <li><code>apic service:list</code> - List the Portal service currently installed.</li> <li><code>-s, --server mgmt_endpoint_url</code>. Use the value of the platform API endpoint URL for this entry.</li> <li><code>--org orgid</code></li> <li><code>--catalog catalog id</code>.</li> <li><code>--debug</code>. If you need debug information.</li> </ul> <p>For more information, see <a href="#">Using the service commands</a>.</p>
<code>apic service-ip-allowlist</code>	Manage your <code>service-ip-allowlist</code> , which enables specific IP addresses to be exempt from being blocked by portal security checks.	<ul style="list-style-type: none"> <li><code>apic service-ip-allowlist:add</code> - Add specific IP addresses to the <code>allowlist</code>.</li> <li><code>apic service-ip-allowlist:delete</code> - Delete all of the IP addresses from the <code>allowlist</code>.</li> <li><code>apic service-ip-allowlist:list</code> - List all of the IP addresses that are currently on the <code>allowlist</code>.</li> <li><code>apic service-ip-allowlist:remove</code> - Remove one or more specific IP addresses from the <code>allowlist</code>.</li> </ul> <p>For more information, see <a href="#">Using the service-ip-allowlist commands</a>.</p>
<code>apic site</code>	Manage your <code>site</code> .	<ul style="list-style-type: none"> <li><code>site:cache-rebuild</code> - Rebuild the cache of a portal site.</li> <li><code>apic site:check</code> - Run a platform check against a portal site.</li> <li><code>apic site:state</code> - Obtain the current state of a portal site.</li> <li><code>apic site:login-link</code> - Gets an admin login link for a specific portal site.</li> <li><code>-s, --server management_server</code>.</li> <li><code>--org orgid</code></li> <li><code>--catalog catalog id</code>.</li> <li><code>--format format</code>. The options are <code>json</code> or <code>yaml</code>.</li> </ul> <p>For more information, see <a href="#">Using the site commands</a>.</p>

Command	Description	Subcommands
<code>apic site-config</code>	Manage your <code>site-config</code> .	<ul style="list-style-type: none"> <li><code>apic site-config:create-export</code> - Create an export of a <code>site config</code>.</li> <li><code>apic site-config:create-import</code> - Create an import of a <code>site config</code>.</li> <li><code>apic site-config:delete-export</code> - Delete an export of a <code>site config</code>.</li> <li><code>apic site-config:delete-import</code> - Delete an import of a <code>site config</code>.</li> <li><code>apic site-config:get-export</code> - Get an export of a <code>site config</code>.</li> <li><code>apic site-config:get-export-status</code> - Get the status of an export of a <code>site config</code>.</li> <li><code>apic site-config:get-import-status</code> - Get the status of an import of a <code>site config</code>.</li> </ul> <p>For more information, see <a href="#">Using the site-config commands</a>.</p>
<code>apic sites</code>	Manage your <code>sites</code> .	<ul style="list-style-type: none"> <li><code>apic sites:check</code> - Run a platform check against all portal sites on a portal service.</li> <li><code>apic sites:list</code> - List the sites of a portal service.</li> </ul> <p><code>-s, --server management_server.</code>  <code>--portal_service_name portal_service_name</code>  <code>--format format</code>. The options are <code>json</code> or <code>yaml</code>.</p> <p>For more information, see <a href="#">Using the sites commands</a>.</p>
<code>apic themes</code>	Manage your <code>themes</code> .	<ul style="list-style-type: none"> <li><code>apic themes:delete</code> - Delete <code>themes</code>.</li> <li><code>apic themes:disable</code> - Disable <code>themes</code>.</li> <li><code>apic themes:enable</code> - Enable <code>themes</code>.</li> <li><code>apic themes:list</code> - Get a list of your <code>themes</code>.</li> <li><code>apic themes:set-default</code> - Set the default theme of the portal.</li> </ul> <p>For more information, see <a href="#">Using the themes commands</a>.</p>
<code>apic twig</code>	Manage the <code>twig</code> debugging operations on a specific portal site.	<ul style="list-style-type: none"> <li><code>apic twig:debug-enable</code> - Enable twig debugging for a portal site.</li> <li><code>apic twig:debug-disable</code> - Disable twig debugging for a portal site.</li> <li><code>apic twig:debug-status</code> - Get the current state of twig debugging for a specific site.</li> </ul> <p>For more information, see <a href="#">Using the twig commands</a>.</p>
<code>apic version</code>	Get the version of your API Connect toolkit.	<ul style="list-style-type: none"> <li><code>apic version --mode portaladmin</code> - Get toolkit version.</li> </ul>

- [Using the api commands](#)  
You can use the `api` commands to get and list the APIs on your Developer Portal service.
- [Using the apic-config commands](#)  
You can use the `apic-config` commands to get the apic config of your Developer Portal service.
- [Using the application commands](#)  
You can use the `application` commands to get and list the applications on your Developer Portal service.
- [Using the backups commands](#)  
You can use the `backups` commands to list the backups on your Developer Portal service.
- [Using the consumer-org commands](#)  
You can use the `consumer-org` commands to get and list the consumer orgs on your Developer Portal service.
- [Using the content commands](#)  
You can use the `content` commands to list and export content from your Developer Portal site.
- [Using the custom-module commands](#)  
You can use the `custom-module` commands to create exports and imports, delete exports and imports, get the status of an export or import, and get an export, on your Developer Portal service.
- [Using the custom-theme commands](#)  
You can use the `custom-theme` commands to create export and import tasks, delete export and import tasks, get the status of an export or import task, and get an export artifact, on your Developer Portal service.
- [Using the custom-translation commands](#)  
You can use the `custom-translation` commands to create an import or export task that will import or export a .tgz of custom translations for a Developer Portal site. You can also delete the export and import tasks, get the status of an export or import task, and get a custom translation export artifact.
- [Using the custom-webserver-page commands](#)  
You can use the `custom-webserver-page` commands to customize the web server error pages on your Developer Portal service.
- [Using the drupal-config commands](#)  
You can use the `drupal-config` commands to list, get the values of, set values, create new, and delete the Drupal configuration objects on your Developer Portal service.
- [Using the drupal-state commands](#)  
You can use the `drupal-state` commands to store and retrieve information about the state of your Developer Portal system.
- [Using the entity commands](#)  
You can use the `entity` commands to get the entity count on your Developer Portal service.
- [Using the factory-reset command](#)  
You can use the `factory-reset` command to perform a factory reset of your Developer Portal, which will delete the portal service and all of its associated Developer Portal sites.
- [Using the forums commands](#)  
You can use the `forums` commands to disable or enable the forums on your Developer Portal service.
- [Using the ip-security-enabled command](#)  
You can use the `ip-security-enabled` commands to toggle IP security on your Developer Portal. When IP security is enabled, modules such as the Drupal Perimeter Defence module, or flood control, will block client IP addresses suspected of malicious behavior, as expected. Banned IP addresses can be cleared by using the

`security:clear-bans` command. You might want to turn off IP security if you are performing penetration tests, or if you cannot pass through the client IP address from your external load balancer. Note that IP security is enabled by default on the Developer Portal.

- [Using the modules commands](#)  
You can use the `modules` commands to list, enable, disable and delete the modules on your Developer Portal service.
- [Using the php-memory commands](#)  
You can use the `php-memory` commands to list or update the PHP memory limits on your Developer Portal.
- [Using the platforms commands](#)  
You can use the `platforms` commands to list the platforms on your Developer Portal service.
- [Using the product commands](#)  
You can use the `product` commands to get and list the products on your Developer Portal service.
- [Using the security command](#)  
You can use the `security` command to clear all of the banned user/IP addresses on a Developer Portal site.
- [Using the service commands](#)  
You can use the `service` commands to check and list the Developer Portal service from the Developer Portal database.
- [Using the service-ip-allowlist commands](#)  
You can use the `service-ip-allowlist` commands to add, remove, and list the IP addresses on your Developer Portal `allowlist`. IP addresses that are on the `allowlist` are exempt from being blocked by Developer Portal security checks, for example, load balancer and proxy IPs.
- [Using the site commands](#)  
You can use the `site` commands to check the site on your Developer Portal service.
- [Using the site-config commands](#)  
You can use the `site-config` commands to create exports and imports, delete exports and imports, get the status of an export or import, and get an export, on your Developer Portal service.
- [Using the sites commands](#)  
You can use the `sites` commands to check and list the sites on your Developer Portal service.
- [Using the themes commands](#)  
You can use the `themes` commands to list, enable, disable, delete, and set the default themes on your Developer Portal service.
- [Using the twig commands](#)  
You can use the `twig` commands to enable, disable, and check the state of twig debugging on a specific site on your Developer Portal service.

---

## Using the api commands

You can use the `api` commands to get and list the APIs on your Developer Portal service.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details of the `apic login` command, see [Logging in to a management server](#).

For a summary of the general-purpose commands and their use, see [Developer Portal CLI commands](#).

2. List the APIs from a site.

For example,

```
apic --mode portaladmin api:list --org orgid/name --server management_server --catalog catalogid/name
```

- `management_server` is the endpoint URL of the management server (required).
- `catalogid/name` is the ID or name of the catalog that the site belongs to (required).
- `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
- `format_type` is the output format. Can be `json`, `yaml`, `go-template=...`, `go-template-file=...`. Defaults to `yaml`.

3. Get an API from a site by using the `id` or `name:version`.

For example,

```
apic --mode portaladmin api:get --org orgid/name --server management_server --catalog catalogid/name id/name:version
```

- `id/name:version` - The ID or name:version of a specific API (required). For example, `id-of-api-called-example-3` or `example:3.0.0`.

4. Get a specific entire API document from the portal of the provided organization and catalog by using the `id` or `name:version`.

```
apic --mode portaladmin api:get-document --org orgid/name --server management_server --catalog catalogid/name --format format_type id/name:version
```

For example,

```
apic --mode portaladmin api:get-document --org ibm --server management_server --catalog portal-test --format yaml intuiz-api:1.0.0
```

5. Add an attachment for your API within your Developer Portal.

For example,

```
apic --mode portaladmin api:add-attachment -s management_server --org orgid/name --catalog catalogid/name --
attachment_name attachment.txt --attachment_description "API documentation" mortgage-management-api:1.0.0 ./attachment.txt
Loading File (Large files may take a while)...
```

```
Attachment successfully added to api mortgage-management-api:1.0.0.
This product now has 4 attachments.
```

- `attachment_name` is the name given to the attachment when it is uploaded (required).
- `attachment_description` is the description of the attachment that is displayed to the users.

6. Set an icon for your API within your Developer Portal.

For example,

```
apic --mode portaladmin api:set-icon -s management_server --org orgid/name --catalog catalogid/name --icon_description
"API icon" mortgage-management-api:1.0.0 ./icon.png
Loading File (Large files may take a while)...
```

```
Icon successfully set for api mortgage-management-api:1.0.0
```

- `icon_description` is the icon description to be displayed to users and it is used as an alt text for the image (required).

7. Add a tag (Category) for your API within your Developer Portal.

For example,

```
apic --mode portaladmin api:add-tag -s management_server --org orgid/name --catalog catalogid/name --tag_name
top_level_element/next_level_element/lower_level_element mortgage-management-api:1.0.0
Successfully set taxonomy tag top_level_element/next_level_element/lower_level_element
for api mortgage-management-api:1.0.0
```

- `tag_name` is the tag name. For example, `top_level_element/next_level_element` (required).

---

## Using the apic-config commands

You can use the **apic-config** commands to get the apic config of your Developer Portal service.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password
provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details of the `apic login` command, see [Logging in to a management server](#).

2. Get the apic config of a site.

For example,

```
apic --mode portaladmin apic-config:get --org orgid/name --server management_server --catalog name/id
```

---

## Using the application commands

You can use the **application** commands to get and list the applications on your Developer Portal service.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password
provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
```

```
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`. For full details of the `apic login` command, see [Logging in to a management server](#).

2. List the applications on a site.  
For example,

```
apic --mode portaladmin application:list --org orgid/name --server management_server --catalog name/id
```

3. Get an application from a site by using application id.  
For example,

```
apic --mode portaladmin application:get --org orgid/name --server management_server --catalog name/id id
```

---

## Using the backups commands

You can use the **backups** commands to list the backups on your Developer Portal service.

1. Log in as Cloud manager:

```
apic login --server management_server --realm admin/identity_provider --username cloud_username --password cloud_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`. For full details of the `apic login` command, see [Logging in to a management server](#).

2. Get the availability zone of the portal service's backups that you want to list:

```
apic availability-zones --org orgid/name --server management_server
```

3. List all the portal services in that availability zone

```
apic portal-services:list --org orgid/name --server management_server --availability-zone availability_zone
```

4. Get the selected portal service's name

```
apic portal-services:get --org orgid/name --server management_server --availability-zone availability_zone PORTAL-SERVICE-NAME
```

5. List the backups of the portal service

```
apic --mode portaladmin backups:list --server management_server --portal_service_name PORTAL-SERVICE-NAME --format json
```

---

## Using the consumer-org commands

You can use the **consumer-org** commands to get and list the consumer orgs on your Developer Portal service.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
```



```
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details of the `apic login` command, see [Logging in to a management server](#).

2. List the consumer orgs of a site.

For example,

```
apic --mode portaladmin consumer-org:list --org orgid/name --server management_server --catalog name/id
```

3. Get a consumer org from a site by using org id.

For example,

```
apic --mode portaladmin consumer-org:get --org orgid/name --server management_server --catalog name/id id
```

---

## Using the content commands

You can use the `content` commands to list and export content from your Developer Portal site.

1. Log in as Provider org as seen in the following example.

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details of the `apic login` command, see [Logging in to a management server](#).

For a summary of the general-purpose commands and their use, see [Developer Portal CLI commands](#).

2. List the content entity types that are exportable from your site by using the `content:list-types` command. This command returns a list of all of the content entity types that are allowed to be exported from your site. If a type isn't listed in the return, that type is either blocked or isn't a content entity type. Each content entity type can have sub-types, which are known as bundles. These bundles can be used in the `content:list` command, when you want to list a particular bundle type of a content entity. In the following example, `page` is a bundle type of the content entity type `node`.

For example,

```
apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name content:list-types
block_content:
- basic
comment:
- comment
- comment_forum
export_entity: { }
file: { }
import_entity: { }
media: { }
menu_link_content: { }
node:
- article
- blog_post
- faq
- forum
- page
shortcut:
- default
taxonomy_term:
- forums
- tags
```

Where:

- `management_server` is the endpoint URL of the management server (required).
- `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
- `catalogid/name` is the ID or name of the catalog that the site belongs to (required).

3. List all of the entities in your site for a given content entity type and bundle by using the `content:list` command.

For example,

```
apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name content:list --entity_type entity_type_name --bundle bundle_name
```

Where:

- `entity_type_name` is the type of entity that you want to list (required). You can use the `content:list-types` command to find the list of available content entity types.
- `bundle_name` is the name of the bundle type that you want to list.

The following example lists the content of the `page` bundle type for the `node` content entity.

```
apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name content:list --entity_type node --bundle page
-
 Title: 'Terms of use'
 Langcode: en
 'Entity ID': '1'
 Bundle: page
 UUID: 75fc430e-fc4d-47e3-bc5f-7f23063716ac
-
 Title: 'Privacy Policy'
 Langcode: en
 'Entity ID': '2'
 Bundle: page
 UUID: ddf300b-16fe-4d1e-be6f-ce7040131b68
```

---

## Using the custom-module commands

You can use the **custom-module** commands to create exports and imports, delete exports and imports, get the status of an export or import, and get an export, on your Developer Portal service.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details of the `apic login` command, see [Logging in to a management server](#).

2. Export your custom modules.

The **custom-module:create-export** command creates an export task against the portal of the specified `catalog` and `org`. The command then polls the status of the task until it has **FINISHED** and the artifacts are streamed back.

Note: The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the commands **custom-module:get-export-status** and **custom-module:get-export** to get the status and potentially an artifact.

```
apic --mode portaladmin custom-module:create-export --catalog name/id --org orgid/name --server management_server
```

- The module `.tgz` is saved to the directory in which you ran the command.
- The saved `.tgz` has the format `custom_module_createExport-20200217134637.tgz`.
- You can save your exports into your change management system.

For example,

```
apic --mode portaladmin custom-module:create-export --catalog dev --org ibm --server api.stagingexample.com
```

3. Import your custom modules.

The **custom-module:create-import** command uses the supplied `.tgz` file to import the custom modules into the portal of the specified `catalog` and `org`. The command then polls the status of the task until the task has either **FINISHED** successfully, or failed because of an error.

Note:

- The higher level folders of the archive can contain only these characters, a-z and 0-9.
- The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the command **custom-module:get-import-status** to get the status of the import.
- If you import a custom module that already exists you overwrite the existing custom module.

```
apic --mode portaladmin custom-module:create-import --catalog name/id --org orgid/name --server management_server /PATH/TO/MODULE/TGZ
```

For example,

```
apic --mode portaladmin custom-module:create-import --catalog prod --org ibm --server api.productionexample.com C:/users/example/desktop/custom_module_createExport-20200217134637.tgz
```

4. You can run the delete command to cancel any ongoing or recently created import or export tasks. Any temporarily created artifacts will be deleted from the portal filesystem. No custom modules are deleted.

For example,

```
apic --mode portaladmin custom-module:delete-export --catalog name/id --org orgid/name --server management_server --format json --task_id string

apic --mode portaladmin custom-module:delete-import --catalog name/id --org orgid/name --server management_server --format json --task_id string
```

---

## Using the custom-theme commands

You can use the **custom-theme** commands to create export and import tasks, delete export and import tasks, get the status of an export or import task, and get an export artifact, on your Developer Portal service.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details of the `apic login` command, see [Logging in to a management server](#).

2. Export your custom themes.

The **custom-theme:create-export** command creates an export task against the portal of the specified `catalog` and `org`. The command then polls the status of the task until it has **FINISHED** and the artifacts are streamed back.

Note: The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the commands **custom-theme:get-export-status** and **custom-theme:get-export** to get the status and potentially an artifact.

```
apic --mode portaladmin custom-theme:create-export --catalog name/id --org orgid/name --server management_server
```

- The theme `.tgz` is saved to the directory in which you ran the command.
- The saved `.tgz` has the format `custom_theme_createExport-20200217134637.tgz`.
- You can save your exports into your change control management system.

For example,

```
apic --mode portaladmin custom-theme:create-export --catalog dev --org ibm --server api.stagingexample.com
```

3. Import your custom themes.

The **custom-theme:create-import** command uses the supplied `.tgz` file to import the custom themes into the portal of the specified `catalog` and `org`. The command then polls the status of the task until the task has either **FINISHED** successfully, or failed because of an error.

Note:

- The higher level folders of the archive can contain only these characters, a-z and 0-9.
- The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the command **custom-theme:get-import-status** to get the status of the import.
- If you import a custom theme that already exists you overwrite the existing custom theme.

```
apic --mode portaladmin custom-theme:create-import --catalog name/id --org orgid/name --server management_server /PATH/TO/theme/TGZ
```

For example,

```
apic --mode portaladmin custom-theme:create-import --catalog prod --org ibm --server api.productionexample.com C:/users/example/desktop/custom_theme_createExport-20200217134637.tgz
```

4. You can run the delete command to cancel any ongoing or recently created import or export tasks. Any temporarily created artifacts will be deleted from the portal filesystem. No custom themes are deleted.

For example,

```
apic --mode portaladmin custom-theme:delete-export --catalog name/id --org orgid/name --server management_server --format json --task_id string

apic --mode portaladmin custom-theme:delete-import --catalog name/id --org orgid/name --server management_server --format json --task_id string
```

---

## Using the custom-translation commands

You can use the **custom-translation** commands to create an import or export task that will import or export a .tgz of custom translations for a Developer Portal site. You can also delete the export and import tasks, get the status of an export or import task, and get a custom translation export artifact.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details of the `apic login` command, see [Logging in to a management server](#).

2. Export your custom translations.

The **custom-translation:create-export** command creates an export task against the portal of the specified `catalog` and `org`, to export an archive of the custom translations of the site. The command then polls the status of the task until it has **FINISHED** and the artifacts are streamed back. The task ID is printed to the terminal.

Note:

- The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the commands **custom-translation:get-export-status** and **custom-translation:get-export** to get the status and potentially an artifact.
- The translation .tgz archive is saved to the directory in which you ran the command.
- The saved .tgz has the format `custom_translation_createExport-20210217134637.tgz`.
- You can save your exports into your change control management system.

```
apic --mode portaladmin custom-translation:create-export --catalog catalogid/name --org orgid/name --server management_server
```

- `catalogid/name` is the ID or name of the catalog that the site belongs to (required).
- `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
- `management_server` is the endpoint URL of the management server (required).
- `--langcodes` is a comma separated list of the language codes, for example `es,zh-hans`. You can use this option to specify which custom translations to export.
- `--no-poll` can be used to stop the command from polling the task, and just return the task ID.

For example:

```
apic --mode portaladmin custom-translation:create-export --catalog dev --org ibm --server api.stagingexample.com
```

3. Import your custom translations.

The **custom-translation:create-import** command uses the supplied .tgz file to create a task to import the custom translations into the portal of the specified `catalog` and `org`. The command then polls the status of the task until the task has either **FINISHED** successfully, or failed because of an error. The task ID is printed to the terminal.

Note:

- The .tgz archive must contain only .po translation files, and nothing else.
- The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the command **custom-translation:get-import-status** to get the status of the import.
- If you import a custom translation that already exists, the existing custom translation is overwritten.

```
apic --mode portaladmin custom-translation:create-import --catalog catalogid/name --org orgid/name --server management_server /PATH/TO/MODULE/TGZ
```

- `--no-poll` can be used to stop the command from polling the task, and just return the task ID.

For example:

```
apic --mode portaladmin custom-translation:create-import --catalog prod --org ibm --server api.productionexample.com C:/users/example/desktop/custom_translation_createExport-20210217134637.tgz
```

4. You can run the delete command to cancel any ongoing or recently created import or export tasks. Any temporarily created artifacts will be deleted from the portal filesystem. No custom translations are deleted.

For example,

```
apic --mode portaladmin custom-translation:delete-export --catalog catalogid/name --org orgid/name --server management_server --format json --task_id taskid
```

```
apic --mode portaladmin custom-translation:delete-import --catalog catalogid/name --org orgid/name --server management_server --format json --task_id taskid
```

- `taskid` is the ID of the import or export task that was created on the queue (required).
- `--format` is the output format that can be seen if you run the command with debug. Can be `json`, `yaml`, `go-template=...`, `go-template-file=...`. Defaults to `yaml`.

For example:

```
apic --mode portaladmin custom-translation:delete-import --catalog prod --org ibm --server api.productionexample.com --task_id 4hythbptta6z9lyx
```

## Using the custom-webserver-page commands

You can use the **custom-webserver-page** commands to customize the web server error pages on your Developer Portal service.

The **custom-webserver-page** commands enable you to set, get, and delete custom HTML content for the web server error pages of index, 404, 40x, and 50x. You can embed base64 encoded images, such as .svg and .png, or inline .svg images in your content, but the maximum size of the content of each error page is 8 MB.

Note:

- The custom web server error pages that can be set by using the **custom-webserver-page** commands, are the pages that are returned when the URL of a specific portal site isn't used. For example, if your portal site was `https://example.com/myorg/dev`, but you went to `https://example.com/myorg/deb`, then this is when a custom 404 error page is returned. However, if the requested URL matches a portal site, then the web server will use the Drupal error handlers, and will respond according to the Drupal configuration.
- All custom content that you set must conform to the HTML5 specification. If you try to set custom content that doesn't conform to HTML5, it will fail with the following error message:

```
Error: The supplied custom HTML failed validation.
```

- Setting and deleting custom web server pages can take several minutes to complete, depending on the size of the content.

1. Log in as Cloud manager:

```
apic login --server management_server --realm admin/identity_provider --username admin --password cloud_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`.

For full details of the `apic login` command, see [Logging in to a management server](#).

2. Set a custom HTML web server error page on the Developer Portal. The maximum size of the page is 8 MB, and you must ensure that the content conforms to the HTML5 specification, or the **custom-webserver-page:set** command will fail with a validation error.

```
apic --mode portaladmin custom-webserver-page:set --server management_server --portal_service_name portal --page_type page_type filename.html
```

- `management_server` is the endpoint URL of the management server (required).
- `portal` is the name of the portal service (required).
- `page_type` is the type of web server error page that you want to run the command against (required). Valid values are `index`, `404`, `40x`, or `50x`.
- `filename` is the name of the HTML file that contains the content for the custom web server error page.

For example, the following screen capture shows a standard 404 web server response:



### 404 Not Found

Then the following **custom-webserver-page:set** command is run:

```
apic --mode portaladmin custom-webserver-page:set --server my.management.server.com --portal_service_name portal --page_type 404 custom-404.html
Loading File (Large files may take a while)...
```

```
Successfully added the custom HTML content for page type 404.
```

The following custom 404 error page is now displayed:

**Error 404**

Hmm, we can't seem to find that.

3. Get the custom content of a specific page type and save it to a file named <page\_type>-timestamp.html.

```
apic --mode portaladmin custom-webserver-page:get --server management_server --portal_service_name portal --page_type page_type
```

For example:

```
apic --mode portaladmin custom-webserver-page:get --server my.management.server.com --portal_service_name portal --page_type 404
Incoming project saved to 404-20211027121830.html
```

If there is no custom content available for the page type, the following error is displayed:

```
apic --mode portaladmin custom-webserver-page:get --server my.management.server.com --portal_service_name portal --page_type 50x
Error: There is currently no custom HTML content for page type 50x.
```

4. Delete the custom content of a specific page type. If there is no custom content available for a page type, the error response defaults to the standard web server error page.

```
apic --mode portaladmin custom-webserver-page:delete --server management_server --portal_service_name portal --page_type page_type
```

For example:

```
apic --mode portaladmin custom-webserver-page:delete --server my.management.server.com --portal_service_name portal --page_type 404
The custom HTML content for page type 404 has been deleted.
```

## Using the drupal-config commands

You can use the **drupal-config** commands to list, get the values of, set values, create new, and delete the Drupal configuration objects on your Developer Portal service.

For example, you can use the **drupal-config** commands to disable CSS and JS aggregation to help with twig debug, see [Example](#).

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password provider_password
```

You can determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is **default-idp-2**.

For full details of the **apic login** command, see [Logging in to a management server](#).

2. List all of the available Drupal config for a site.

```
apic --mode portaladmin drupal-config:list --server management_server --catalog catalogid/name --org orgid/name --format format_type
```

- `management_server` is the endpoint URL of the management server (required).
- `catalogid/name` is the ID or name of the catalog that the site belongs to (required).
- `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
- `format_type` is the output format. Can be `json`, `yaml`, `go-template=...`, `go-template-file=...`. Defaults to `yaml`.

For example:

```
apic --mode portaladmin drupal-config:list --server my.management.server.com --org ibm --catalog portal-test --format yaml
admin_toolbar_tools.settings:
 name: admin_toolbar_tools.settings
adminimal_admin_toolbar.settings:
 name: adminimal_admin_toolbar.settings
advagg.settings:
 name: advagg.settings
advagg_css_minify.settings:
 name: advagg_css_minify.settings
advagg_js_minify.settings:
 name: advagg_js_minify.settings
autologout.settings:
 name: autologout.settings
...
```

Note that the example has been reduced as the full response is very large.

3. List a specific Drupal config for a site.

```
apic --mode portaladmin drupal-config:list --server management_server --catalog catalogid/name --org orgid/name --prefix config_prefix --format format_type
```

- `config_prefix` is the prefix of the Drupal config, for example `system`. If omitted, the command returns all config names in the system.

For example:

```
apic --mode portaladmin drupal-config:list --server my.management.server.com --org ibm --catalog portal-test --format yaml
--prefix system.theme
system.theme:
 name: system.theme
system.theme.global:
 name: system.theme.global
```

4. Update a Drupal config key value.

```
apic --mode portaladmin drupal-config:set --server management_server --catalog catalogid/name --org orgid/name --config_name config_name --config_key config_key --config_value config_value
```

- `config_name` is the name of the Drupal config object, for example `system.site` (required).
- `config_key` is the Drupal config key, for example `page.front` (required).
- `config_value` is the value to assign to the config key (required).

For example, to set the default theme to `claro`:

```
apic --mode portaladmin drupal-config:set --server my.management.server.com --org ibm --catalog portal-test --config_name system.theme --config_key default --config_value claro
// Do you want to update default key in system.theme config?: yes.
```

5. Create a new Drupal config key and set the value.

```
apic --mode portaladmin drupal-config:set --server management_server --catalog catalogid/name --org orgid/name --config_name config_name --config_key new_config_key --config_value new_config_value
```

For example, to create a new config key and set the value in the `system.site` config object:

```
apic --mode portaladmin drupal-config:set --server my.management.server.com --org ibm --catalog portal-test --config_name system.site --config_key new_key --config_value new_value
// new_key key does not exist in system.site config. Do you want to create a new config key?: yes.
```

6. Get all of the key values for a Drupal config object.

```
apic --mode portaladmin drupal-config:get --server management_server --catalog catalogid/name --org orgid/name --format format_type --config_name config_name
```

For example, to find out what themes are available:

```
apic --mode portaladmin drupal-config:get --server my.management.server.com --org ibm --catalog portal-test --format json
--config_name system.theme
{
 "admin": "seven",
 "default": "claro",
 "_core": {
 "default_config_hash": "fOjer9hADYYnbCJVZMFZIIIMlazTFWyg84ZkFDHfAbUg"
 },
 "defaultplus": "bartisk"
}
```

7. Get the value of a specific key for a Drupal config object.

```
apic --mode portaladmin drupal-config:get --server management_server --catalog catalogid/name --org orgid/name --format format_type --config_name config_name --config_key config_key
```

For example, to find out what the default theme is:

```
apic --mode portaladmin drupal-config:get --server my.management.server.com --org ibm --catalog portal-test --format json
--config_name system.theme --config_key default
{
```

```
 "system.theme.default": "claro"
 }
}
```

8. Delete a Drupal config object, or a specific config key and its value.

```
apic --mode portaladmin drupal-config:delete --server management_server --catalog catalogid/name --org orgid/name --
config_name config_name --config_key config_key
```

- `config_key` is the Drupal config key, for example `page.front` (optional).

For example, to delete the config key `new_key` and its value:

```
apic --mode portaladmin drupal-config:delete --server my.management.server.com --org ibm --catalog portal-test --
config_name system.site --config_key new_key
The config object/value was successfully deleted.
```

## Example

The following scenario gives an example of how to turn off CSS and JS aggregation to help with twig debug. However, on a production system aggregation should be enabled for performance reasons.

1. Check the settings for performance, which include settings for CSS and JS aggregation:

```
apic --mode portaladmin drupal-config:get --server management_server --catalog catalogid/name --org orgid/name --
config_name system.performance
```

2. Check the settings for advanced aggregation:

```
apic --mode portaladmin drupal-config:get --server management_server --catalog catalogid/name --org orgid/name --
config_name advagg.settings
```

3. Turn off CSS aggregation:

```
apic --mode portaladmin drupal-config:set --server management_server --catalog catalogid/name --org orgid/name --
config_name system.performance --config_key css.preprocess --config_value 0
```

4. Turn off JS aggregation:

```
apic --mode portaladmin drupal-config:set --server management_server --catalog catalogid/name --org orgid/name --
config_name system.performance --config_key js.preprocess --config_value 0
```

5. Turn off advanced aggregation (this setting is needed for the previous aggregation settings to apply):

```
apic --mode portaladmin drupal-config:set --server management_server --catalog catalogid/name --org orgid/name --
config_name advagg.settings --config_key enabled --config_value 0
```

6. Then, rebuild the cache:

```
apic --mode portaladmin site:cache-rebuild --server management_server --catalog catalogid/name --org orgid/name
```

When loading the site in a browser, you should now see CSS and JS aggregation is disabled when using development tools for site debug or customization.

Note:

- Disabling aggregation affects site performance, so if the site goes into production remember to turn aggregation back on again.
- Although the settings are shown as `true` or `false`, to set them by using the `drupal-config:set` command, you must apply a value of 1 for `true`, and 0 for `false`.

## Using the drupal-state commands

You can use the **drupal-state** commands to store and retrieve information about the state of your Developer Portal system.

Information about the state of your Developer Portal system is stored in the Developer Portal database, and therefore this information is lost if the database is dropped, or the Developer Portal site is re-installed from configuration.

The **drupal-state** commands can be used to set, update, get, or delete a state key value. For example, you can use the **drupal-state** commands to enable and disable maintenance mode for your Developer Portal; for more information, see the scenario [How to enable and disable maintenance mode on your Developer Portal](#).

Warning: Use the **drupal-state** command with caution. This command enables you to modify keys for your Developer Portal site state. Incorrect usage of this command can bring down your Developer Portal, and then require restoring it from a backup.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password
provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```



The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`. For full details of the `apic login` command, see [Logging in to a management server](#).

2. Create a new Drupal state key, or update an existing key, and set the value.

```
apic --mode portaladmin drupal-state:set --server management_server --catalog catalogid/name --org orgid/name --input_format input_format_type --state_key state_key_name --state_value state_value
```

Where:

- `management_server` is the endpoint URL of the management server (required).
- `catalogid/name` is the ID or name of the catalog that the site belongs to (required).
- `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
- `input_format_type` is the input format of the value for the state key. Can be `string`, `integer`, `float`, `boolean`, `json`, or `yaml`. Defaults to `auto`, which means that you can omit this parameter, and the toolkit CLI will decide which format to use.
- `state_key_name` is the name of the state key, for example `system.cron_last` (required).
- `state_value` is the value to assign to the state key (required).
- `--format` is the output format. Can be `json`, `yaml`, `go-template=...`, `go-template-file=...`. Defaults to `yaml`.

For example, to create a new key called `testKey`, and set the value to `123`:

```
apic --mode portaladmin drupal-state:set -s my.management.server.com -o myorg -c dev-catalog --input_format yaml --state_key testKey --state_value 123
Successfully set state testKey to the value of 123
```

3. Get the value for a particular Drupal state key.

```
apic --mode portaladmin drupal-state:get --server management_server --catalog catalogid/name --org orgid/name --format format_type --state_key state_key_name
```

For example, to get the value of the state key `testKey`:

```
apic --mode portaladmin drupal-state:get -s my.management.server.com -c dev-catalog -o myorg --format yaml --state_key testKey
testKey: 123
```

4. Delete a specific Drupal state key and its value.

```
apic --mode portaladmin drupal-state:delete --server management_server --catalog catalogid/name --org orgid/name --state_key state_key_name
```

For example, to delete the state key `testKey` and its value:

```
apic --mode portaladmin drupal-state:delete -s my.management.server.com -c dev-catalog -o myorg --state_key testKey
The key/value was successfully deleted.
```

## Related tasks

- [Toggling the site in and out of maintenance mode](#)

## Using the entity commands

You can use the **entity** commands to get the entity count on your Developer Portal service.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`. For full details of the `apic login` command, see [Logging in to a management server](#).

2. Get the entity count for a site.

For example,

```
apic --mode portaladmin entity:count --org orgid/name --server management_server --catalog name/id
```

---

## Using the factory-reset command

You can use the **factory-reset** command to perform a factory reset of your Developer Portal, which will delete the portal service and all of its associated Developer Portal sites.

Warning: The **factory-reset** command is irreversible. You must ensure that you have backups created before running this command.

1. Log in as Cloud manager:

```
apic login --server management_server --realm admin/identity_provider --username admin --password cloud_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **default-idp-1**.

For full details of the `apic login` command, see [Logging in to a management server](#).

2. Output a dry run status of a Developer Portal factory reset:

```
apic --mode portaladmin factory-reset:delete --server management_server --portal_service_endpoint portal_endpoint
```

- *management\_server* is the endpoint URL of the management server.
- *portal\_endpoint* is the URL endpoint of the portal service.

For example:

```
apic --mode portaladmin factory-reset:delete --server api.stagingexample.com --portal_service_endpoint https://api.portal.mystack.com
```

3. Perform a factory reset of the Developer Portal, deleting the portal service and all of its associated Developer Portal sites:

Warning: The **factory-reset** command is irreversible. You must ensure that you have backups created before running this command.

```
apic --mode portaladmin factory-reset:delete --server management_server --portal_service_endpoint portal_endpoint --execute_reset true
```

- `--execute_reset` must be set to `true` to trigger the Developer Portal reset.

For example:

```
apic --mode portaladmin factory-reset:delete --server api.stagingexample.com --portal_service_endpoint https://api.portal.mystack.com --execute_reset true
```

---

## Using the forums commands

You can use the **forums** commands to disable or enable the forums on your Developer Portal service.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is **default-idp-2**.

For full details of the `apic login` command, see [Logging in to a management server](#).

2. Disable the Forum module on your Developer Portal site.

This command deletes all of the taxonomy terms in the forum vocabulary, and then disables the Forum module.

```
apic --mode portaladmin forums:disable --org orgid/name --server management_server --catalog name/id
```

- **orgid/name** is the ID or name of the provider organization that the catalog belongs to (required).
- **management\_server** is the endpoint URL of the management server (required).
- **name/id** is the ID or name of the catalog that the site belongs to (required).

For example,

```
apic --mode portaladmin forums:disable --org ibm --server api.stagingexample.com --catalog dev
```

3. Enable the Forum module on your Developer Portal site.

```
apic --mode portaladmin forums:enable --org orgid/name --server management_server --catalog name/id
```

For example,

```
apic --mode portaladmin forums:enable --org ibm --server api.stagingexample.com --catalog dev
```

---

## Using the ip-security-enabled command

You can use the **ip-security-enabled** commands to toggle IP security on your Developer Portal. When IP security is enabled, modules such as the Drupal Perimeter Defence module, or flood control, will block client IP addresses suspected of malicious behavior, as expected. Banned IP addresses can be cleared by using the **security:clear-bans** command. You might want to turn off IP security if you are performing penetration tests, or if you cannot pass through the client IP address from your external load balancer. Note that IP security is enabled by default on the Developer Portal.

1. Log in as Cloud manager:

```
apic login --server management_server --realm admin/identity_provider --username admin --password cloud_password
```

You can determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **default-idp-1**.

For full details of the **apic login** command, see [Logging in to a management server](#).

2. Enable IP security on the Developer Portal:

```
apic --mode portaladmin ip-security-enabled:update --server management_server --portal_service_name portal --enabled true
```

- **management\_server** is the endpoint URL of the management server.
- **portal** is the name of the portal service.
- **--enabled** is set to **true** to enable IP security. Can be set to **false** to disable IP security.

For example:

```
apic --mode portaladmin ip-security-enabled:update --server my.management.server.com --portal_service_name
my_portal_service --enabled true
IP security has been successfully set to true
```

3. Disable IP security on the Developer Portal:

```
apic --mode portaladmin ip-security-enabled:update --server management_server --portal_service_name portal --enabled false
```

---

## Using the modules commands

You can use the **modules** commands to list, enable, disable and delete the modules on your Developer Portal service.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password
provider_password
```

You can determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
```

```
realm: provider/default-idp-2
- title: Corporate LDAP user registry
realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details of the `apic login` command, see [Logging in to a management server](#).

For a summary of the general-purpose commands and their use, see [Developer Portal CLI commands](#).

2. List the modules from a `packagename`.  
For example,

```
apic --mode portaladmin modules:list --org orgid/name --server management_server --catalog name/id --package packagename1,packagename2
```

3. Enable the `modulename` module. You can upload this module by using the `custom-module:create-import` command. For more information, see [Using the custom-module commands](#).

For example,

```
apic --mode portaladmin modules:enable --org orgid/name --server management_server --catalog name/id --modules modulename
```

- `management_server` is the endpoint URL of the management server (required).
- `catalogid/name` is the ID or name of the catalog that the site belongs to (required).
- `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
- `--package` enables you to filter by project packages. You can use multiple comma separated values.
- `--status` enables you to filter by extension status. Can be `enabled`, `disabled` and/or `'not installed'`. You can use multiple comma separated values.
- `--custom` enables you to filter your results to only the modules that you have installed on your Developer Portal site.

4. Disable the `modulename` module. If you disable a module, you must also disable all its dependencies.  
For example,

```
apic --mode portaladmin modules:disable --org orgid/name --server management_server --catalog name/id --modules modulename,module_lib
```

5. Delete the `modulename` module. The module and all its dependencies must be disabled before you can delete them.  
For example,

```
apic --mode portaladmin modules:delete --org orgid/name --server management_server --catalog name/id --modules modulename
```

Note that if you want to manage the Forum module, you can use the `forums` commands. For more information, see [Using the forums commands](#).

---

## Using the php-memory commands

You can use the `php-memory` commands to list or update the PHP memory limits on your Developer Portal.

1. Log in as Cloud manager:

```
apic login --server management_server --realm admin/identity_provider --username cloud_username --password cloud_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`.

For full details of the `apic login` command, see [Logging in to a management server](#).

2. Get the availability zone of the portal service's sites that you want to list:

```
apic availability-zones --org orgid/name --server management_server
```

3. List all the portal services in that availability zone:

```
apic portal-services:list --org orgid/name --server management_server --availability-zone availability_zone
```

4. Get the selected portal service's name:

```
apic portal-services:get --org orgid/name --server management_server --availability-zone availability_zone PORTAL-SERVICE-NAME
```

5. List the current PHP memory limits on the Developer Portal platform:

```
apic --mode portaladmin php-memory:list --server management_server --portal_service_name PORTAL-SERVICE-NAME
```

You receive output like this example:

```
Current web memory is set to: 768M
Current admin memory is set to: 768M
```

6. Update the current PHP memory limits on the Developer Portal platform:

```
apic --mode portaladmin php-memory:update --server management_server --portal_service_name PORTAL-SERVICE-NAME --memory_value 1024
```

You receive output like this example:

```
Setting php memory limit to 1024M within the database
Setting php memory limit to 1024M in /var/devportal/storenosync/php.ini
Setting php memory limit to 1024M in /web/local/php_config/php.ini
Done
```

Note: Consider the memory resource limits that are allocated to your Developer Portal pods before you set this value. Setting this value too high can result in the portal pods restarting because the containers request more memory than is allowed by Kubernetes. Setting the value too low can result in the internal portal processes failing because of lack of available memory.

---

## Using the platforms commands

You can use the **platforms** commands to list the platforms on your Developer Portal service.

1. Log in as Cloud manager:

```
apic login --server management_server --realm admin/identity_provider --username cloud_username --password cloud_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`.

For full details of the `apic login` command, see [Logging in to a management server](#).

2. Get the availability zone of the portal service's platforms that you want to list:

```
apic availability-zones --org orgid/name --server management_server
```

3. List all the portal services in that availability zone

```
apic portal-services:list --org orgid/name --server management_server --availability-zone availability_zone
```

4. Get the selected portal service's name

```
apic portal-services:get --org orgid/name --server management_server --availability-zone availability_zone PORTAL-SERVICE-NAME
```

5. List the platforms of the portal service

```
apic --mode portaladmin platforms:list --server management_server --portal_service_name PORTAL-SERVICE-NAME --format json
```

---

## Using the product commands

You can use the **product** commands to get and list the products on your Developer Portal service.

1. Log in as Provider org as seen in the following example.

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
```

```
realm: provider/default-idp-2
- title: Corporate LDAP user registry
realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details of the `apic login` command, see [Logging in to a management server](#).

For a summary of the general-purpose commands and their use, see [Developer Portal CLI commands](#).

2. Get a list of products from a site.

For example,

```
apic --mode portaladmin product:list --org orgid/name --server management_server --catalog catalogid/name
```

- `management_server` is the endpoint URL of the management server (required).
- `catalogid/name` is the ID or name of the catalog that the site belongs to (required).
- `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
- `format_type` is the output format. Can be `json`, `yaml`, `go-template=...`, `go-template-file=...`. Defaults to `yaml`.

3. Get a product from a site by using the `id` or `name:version`.

For example,

```
apic --mode portaladmin product:get --org orgid/name --server management_server --catalog catalogid/name id/name:version
```

- `id/name:version` - The ID or name:version of a specific product (required). For example, `id-of-product-called-example-3` or `example:3.0.0`.

4. Get a specific entire product document from the portal of the provided organization and catalog by using the `id` or `name:version`.

```
apic --mode portaladmin product:get-document --org orgid/name --server management_server --catalog catalogid/name --format
format_type id/name:version
```

For example,

```
apic --mode portaladmin product:get-document --org ibm --server my.management.server.com --catalog portal-test --format
yaml intuiz-product:1.0.0
```

5. Add an attachment for your product within your Developer Portal.

For example,

```
apic --mode portaladmin product:add-attachment -s management_server --org orgid/name --catalog catalogid/name --
attachment_name attachment.txt --attachment_description "Product documentation" awesome-sleek-soft-chips-product:1.0.0
./attachment.txt
Loading File (Large files may take a while)...
```

```
Attachment successfully added to product awesome-sleek-soft-chips-product:1.0.0.
This product now has 2 attachments.
```

- `attachment_name` is the name given to the attachment when it is uploaded (required).
- `attachment_description` is the description of the attachment that is displayed to the users.

6. Set an icon for your product within your Developer Portal.

For example,

```
apic --mode portaladmin product:set-icon -s management_server --org orgid/name --catalog catalogid/name --icon_description
"product icon" awesome-sleek-soft-chips-product:1.0.0 ./icon.png
Loading File (Large files may take a while)...
```

```
Icon successfully set for product awesome-sleek-soft-chips-product:1.0.0
```

- `icon_description` is the icon description to be displayed to users and it is used as an alt text for the image (required).

7. Add a tag (Category) for your product within your Developer Portal.

For example,

```
apic --mode portaladmin product:add-tag -s management_server --org orgid/name --catalog catalogid/name --tag_name
top_level_element/next_level_element/lower_level_element awesome-sleek-soft-chips-product:1.0.0
Successfully set taxonomy tag top_level_element/next_level_element/lower_level_element
for product api-connect-cassin-llc-awesome-sleek-soft-chips-product:1.0.0
```

- `tag_name` is the tag name. For example, `top_level_element/next_level_element` (required).

---

## Using the security command

You can use the `security` command to clear all of the banned user/IP addresses on a Developer Portal site.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password
provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```

apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap

```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is **default-idp-2**.

For full details of the **apic login** command, see [Logging in to a management server](#).

2. Clear all of the user and IP addresses that are currently banned on a particular portal site:

```

apic --mode portaladmin security:clear-bans --server management_server --catalog catalogid/name --org orgid/name

```

- **management\_server** is the endpoint URL of the management server.
- **catalogid/name** is the ID or name of the catalog that the site belongs to.
- **orgid/name** is the ID or name of the provider organization that the catalog belongs to.

For example:

```

apic --mode portaladmin security:clear-bans --server my.management.server.com --catalog portal-test --org ibm
[notice] All bans cleared.

```

Note: You can also manage banned IP addresses by using the Developer Portal UI. For more information, see [Managing banned IP addresses](#).

---

## Using the service commands

You can use the **service** commands to check and list the Developer Portal service from the Developer Portal database.

1. Log in as Cloud manager:

```

apic login --server management_server --realm admin/identity_provider --username cloud_username --password cloud_password

```

You can determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```

apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm

```

For example:

```

apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap

```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **default-idp-1**.

For full details of the **apic login** command, see [Logging in to a management server](#).

For a summary of the general-purpose commands and their use, see [Developer Portal CLI commands](#).

2. Get the availability zone of the portal service's sites that you want to list.

```

apic availability-zones --org orgid/name --server management_server

```

- **apic service:list** - List the Portal service currently installed.
- **-s, --server mgmt\_endpoint\_url**. Use the value of the platform API endpoint URL for this entry.
- **--org orgid**
- **--catalog catalog id**.
- **--debug**. If you need debug information.

3. List the installed Developer Portal service.

```

apic --mode portaladmin service:list -s management_server --portal_service_name PORTAL-SERVICE-NAME --format json

```

---

## Using the service-ip-allowlist commands

You can use the **service-ip-allowlist** commands to add, remove, and list the IP addresses on your Developer Portal **allowlist**. IP addresses that are on the **allowlist** are exempt from being blocked by Developer Portal security checks, for example, load balancer and proxy IPs.

1. Log in as Cloud manager:

```

apic login --server management_server --realm admin/identity_provider --username admin --password cloud_password

```

You can determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```

apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm

```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`. For full details of the `apic login` command, see [Logging in to a management server](#).

2. Add one or more IP addresses to the `allowlist`:

```
apic --mode portaladmin service-ip-allowlist:add --server management_server --portal_service_name portal --ips
"list_of_ips"
```

- `management_server` is the endpoint URL of the management server.
- `portal` is the name of the portal service.
- `list_of_ips` is a comma separated list of the IP addresses that you want to add to the `allowlist`.

For example:

```
apic --mode portaladmin service-ip-allowlist:add --server my.management.server.com --portal_service_name my_portal_service
--ips "123.456.34.56,123.456.46.67"
Successfully added IP: 123.456.34.56.
Successfully added IP: 123.456.46.67.
```

3. Delete all of the IP addresses that are currently on the `allowlist`:

```
apic --mode portaladmin service-ip-allowlist:delete --server management_server --portal_service_name portal
```

4. List all of the IP addresses that are currently on the `allowlist`:

```
apic --mode portaladmin service-ip-allowlist:list --server management_server --portal_service_name portal
```

5. Remove one or more IP addresses from the `allowlist`:

```
apic --mode portaladmin service-ip-allowlist:remove --server management_server --portal_service_name portal --ips
"list_of_ips"
```

---

## Using the site commands

You can use the `site` commands to check the site on your Developer Portal service.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password
provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`. For full details of the `apic login` command, see [Logging in to a management server](#).

For a summary of the general-purpose commands and their use, see [Developer Portal CLI commands](#).

2. Check the site on your portal service. This command checks the file system, database, and API Manager on your portal site, which might identify platform-related problems on your portal site.

```
apic --mode portaladmin site:check --server management_server --catalog catalogid/name --org orgid/name --format
format_type
```

- `management_server` is the endpoint URL of the management server (required).
- `catalogid/name` is the ID or name of the catalog that the site belongs to (required).
- `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
- `format_type` is the output format. Can be `json`, `yaml`, `go-template=...`, `go-template-file=...`. Defaults to `yaml`.

3. Rebuild the cache of a specific portal site.

```
apic --mode portaladmin site:cache-rebuild --server management_server --catalog catalogid/name --org orgid/name
```



For example:

```
apic --mode portaladmin site:cache-rebuild --server management_server --org ibm --catalog portal-test
[success] Cache rebuild complete.
```

4. Display the state of a portal site associated with a particular organization and catalog. This command is useful as a general health check of the site.

```
apic --mode portaladmin site:state --server management_server --catalog catalogid/name --org orgid/name --format
format_type
```

```
apic --mode portaladmin site:state --server my.management.server.com --org ibm --catalog portal-test --format json
{
 "alias": "ibm.portal-test.my.management.server.com",
 "check_state": "OK: 200",
 "content_refresh_required": false,
 "known_to_apim": 1,
 "platform": "platform_devportal_9_x_10_99_99_99_20210922_2000",
 "snapshot_first_requested": "0000-00-00 00:00:00",
 "snapshots_failed_processing_since_last_success": 0,
 "snapshots_requested_since_last_received": 0,
 "state": "INSTALLED",
 "subscribed_to_webhooks": true,
 "url": "my.management.server.com/ibm/portal-test",
 "uuid": "cc0ad766-e372-4988-9d45-744eb8d381de.1cd9dc41-b08d-xyz"
}
```

5. Log in to your Developer Portal admin with a link. This command displays a one time login link for the admin account for your Developer Portal site that is specified by the catalog and organization.

```
apic --mode portaladmin site:login-link --server management_server --org ibm --catalog portal-test
sample output:
https://my.portal.server.com/ibm/portal-test/user/reset/1/12345678901/ABCde1f2gH3ijKL-
4M_NOPqrs5TU6VWxy7ZABCDEFG7h8/new/login
```

---

## Using the site-config commands

You can use the **site-config** commands to create exports and imports, delete exports and imports, get the status of an export or import, and get an export, on your Developer Portal service.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username cloud_username --password
cloud_password
```

You can determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is **default-idp-2**.

For full details of the **apic login** command, see [Logging in to a management server](#).

2. Export your site configs.

The **site-config:create-export** command creates an export task against the portal of the specified **catalog** and **org**. The command then polls the status of the task until it has **FINISHED** and the artifacts are streamed back.

Note: The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the commands **site-config:get-export-status** and **site-config:get-export** to get the status and potentially an artifact.

```
apic --mode portaladmin site-config:create-export --catalog name/id --org orgid/name --server management_server
```

- The site config .tgz is saved to the directory in which you ran the command.
- The saved .tgz has the format **site\_config\_createExport-20200217134637.tgz**.
- You can save your exports into your change control management system.

For example,

```
apic --mode portaladmin site-config:create-export --catalog dev --org ibm --server api.stagingexample.com
```

3. Import your site configs.

The **site-config:create-import** command uses the supplied .tgz file to import the site configs into the portal of the specified **catalog** and **org**. The command then polls the status of the task until the task has either **FINISHED** successfully, or failed because of an error.

Note: The higher level folders of the archive can contain only these characters, a-z and 0-9.

Note: The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the command **site-config:get-import-status** to get the status of the import.

```
apic --mode portaladmin site-config:create-import --catalog name/id --org orgid/name --server management_server
/PATH/TO/SITE_CONFIG/TGZ
```

For example,

```
apic --mode portaladmin site-config:create-import --catalog prod --org ibm --server api.productionexample.com
C:/users/example/desktop/site_config_createExport-20200217134637.tgz
```

4. You can run the delete command to cancel any ongoing or recently created import or export tasks. Any temporarily created artifacts will be deleted from the portal filesystem.

For example,

```
apic --mode portaladmin site-config:delete-export --catalog name/id --org orgid/name --server management_server --format
json --task_id string

apic --mode portaladmin site-config:delete-import --catalog name/id --org orgid/name --server management_server --format
json --task_id string
```

---

## Using the sites commands

You can use the **sites** commands to check and list the sites on your Developer Portal service.

1. Log in as Cloud manager:

```
apic login --server management_server --realm admin/identity_provider --username cloud_username --password cloud_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: admin/default-idp-1
- title: Corporate LDAP user registry
 realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`.

For full details of the `apic login` command, see [Logging in to a management server](#).

2. Get the availability zone of the portal service's sites that you want to list.

```
apic availability-zones --org orgid/name --server management_server
```

3. List all the portal services in that availability zone.

```
apic portal-services:list --org orgid/name --server management_server --availability-zone availability_zone
```

4. Get the selected portal service's name.

```
apic portal-services:get --org orgid/name --server management_server --availability-zone availability_zone PORTAL-SERVICE-
NAME
```

5. Check the sites on your portal service. This command checks the file system, database, and API Manager on all your portal sites, which might identify platform-related problems on one or more of your portal sites.

```
apic --mode portaladmin sites:check --server management_server --portal_service_name PORTAL-SERVICE-NAME --format yaml
```

6. List the sites of the portal service.

```
apic --mode portaladmin sites:list --server management_server --portal_service_name PORTAL-SERVICE-NAME --format json
```

---

## Using the themes commands

You can use the **themes** commands to list, enable, disable, delete, and set the default themes on your Developer Portal service.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password
provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
```

```
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`. For full details of the `apic login` command, see [Logging in to a management server](#).

For a summary of the general-purpose commands and their use, see [Developer Portal CLI commands](#).

2. List the themes from a `packagename`.

For example,

```
apic --mode portaladmin themes:list --org orgid/name --server management_server --catalog catalogid/name --package packagename1,packagename2
```

- `management_server` is the endpoint URL of the management server (required).
- `catalogid/name` is the ID or name of the catalog that the site belongs to (required).
- `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
- `--package` enables you to filter by project packages. You can use multiple comma separated values.
- `--status` enables you to filter by extension status. Can be `enabled`, `disabled` and/or `'not installed'`. You can use multiple comma separated values.
- `--custom` enables you to filter your results to only the themes that you have installed on your Developer Portal site.

3. Enable the `themename` theme. You can upload this theme by using the `custom-theme:create-import` command. For more information, see [Using the custom-theme commands](#).

For example,

```
apic --mode portaladmin themes:enable --org orgid/name --server management_server --catalog catalogid/name --themes themename
```

- `--themes` is the name of the theme, and can be multiple comma separated values (required).

4. Disable the `themename` theme.

For example,

```
apic --mode portaladmin themes:disable --org orgid/name --server management_server --catalog catalogid/name --themes themename,theme_lib
```

5. Delete the `themename` theme. The theme and all its dependencies must be disabled before you can delete them.

For example,

```
apic --mode portaladmin themes:delete --org orgid/name --server management_server --catalog catalogid/name --themes themename
```

6. Set the `themename` theme to be the default theme. The theme must exist and be enabled for it to be set as a default theme.

For example,

```
apic --mode portaladmin themes:set-default --org orgid/name --server management_server --catalog catalogid/name themename
```

- `themename` is the name of the theme to set as default.

---

## Using the twig commands

You can use the `twig` commands to enable, disable, and check the state of twig debugging on a specific site on your Developer Portal service.

Note: Twig debugging should not be left enabled on production sites, as it causes performance issues, and can affect the caching on the site.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`. For full details of the `apic login` command, see [Logging in to a management server](#).

2. Enable twig debugging for a portal site.

```
apic --mode portaladmin twig:debug-enable --server management_server --org orgid/name --catalog catalogid/name
```

- `management_server` is the endpoint URL of the management server (required).
- `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).

- `catalogid/name` is the ID or name of the catalog that the site belongs to (required).

For example:

```
apic --mode portaladmin twig:debug-enable --server my.management.server.com --org ibm --catalog portal-test
Setting twig debug to true for site my.management.server.com/ibm/portal-test.
Rebuilding cache for my.management.server.com/ibm/portal-test so twig debug changes can take affect.
```

3. Disable twig debugging for a portal site.

```
apic --mode portaladmin twig:debug-disable --server management_server --org orgid/name --catalog catalogid/name
```

For example:

```
apic --mode portaladmin twig:debug-disable --server my.management.server.com --org ibm --catalog portal-test
Setting twig debug to false for site my.management.server.com/ibm/portal-test.
Rebuilding cache for my.management.server.com/ibm/portal-test so twig debug changes can take affect.
```

4. Get the current state of twig debugging for a specific site.

```
apic --mode portaladmin twig:debug-status --server management_server --org orgid/name --catalog catalogid/name
```

For example:

```
apic --mode portaladmin twig:debug-status --server my.management.server.com --org ibm --catalog portal-test
The current state of twig debugging is true for my.management.server.com/ibm/portal-test.
```

---

## Scenarios that use the Portal CLI

Here are some scenarios of how you can use the Portal commands in the toolkit CLI to accomplish tasks that are more complex to complete, or not possible in the Developer Portal UI.

- [Exporting and importing custom themes and site configuration](#)  
You can use the portal commands in the toolkit CLI to replicate changes from one Developer Portal instance to another. You might want to use these commands to copy your themes or a configuration change, from a staging environment to a production environment.
- [How to enable and disable maintenance mode on your Developer Portal](#)  
You can use the Portal commands in the toolkit CLI to enable and disable maintenance mode on your Developer Portal site.

---

## Exporting and importing custom themes and site configuration

You can use the portal commands in the toolkit CLI to replicate changes from one Developer Portal instance to another. You might want to use these commands to copy your themes or a configuration change, from a staging environment to a production environment.

---

### Before you begin

To complete this scenario, you must have a staging Developer Portal and a production Developer Portal enabled. You must have administrator access, and the permission `settings:manage`, to both to complete this scenario. The tutorial [Creating the Portal](#) explains how to enable the portal if you have not already done so.

---

### About this scenario

In this scenario, on your staging Developer Portal, you create a theme and make some configuration changes. Then, you use the toolkit CLI to replicate the changes on your production Developer Portal.

This scenario takes you through the following steps:

1. [Create a custom theme on the staging Developer Portal.](#)
2. [Add a field to an application content type to change the site-config on the staging Developer Portal.](#)
3. [Use the cli to export the custom themes and site-config on the staging Developer Portal.](#)
4. [Use the cli to import the custom themes and site-config on the production Developer Portal.](#)

---

## Create a custom theme on the staging Developer Portal

1. Log in to your staging Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click **Manage** to display it.
3. Click **Appearance**, **Generate sub-theme**. The **Generate sub-theme** window is displayed.
4. Enter a Sub-theme name, and select **CSS** for the Sub-theme type. (If you prefer, you can select **SCSS**, but this extension to CSS is for advanced theme developers, and isn't covered by this scenario.)

## Generate sub-theme ☆

[List](#)[Generate](#)[Update](#)[Settings](#)

[Home](#) » [Administration](#)

The first step in customizing the branding of your Developer Portal is to create a custom sub-theme.

The sub-theme inherits all of the resources of the parent theme, and you can then override specific resources in the overrides.css file to configure your customizations. For more information, see: [Knowledge Center](#)

Complete the form below and you will be presented with a custom sub-theme to download.

Sub-theme name \*

A custom theme name, for example: 'mycustom\_theme' or 'banka\_theme'. The name does not need to end in '\_theme' but it is a common convention.

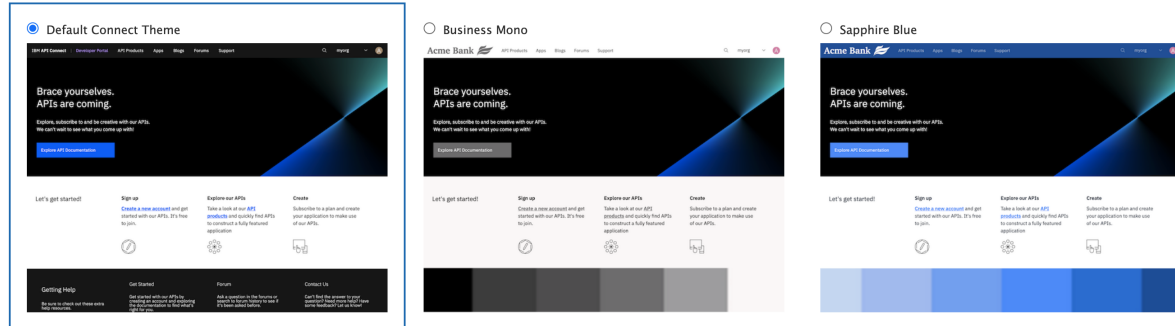
Sub-theme type

CSS

SCSS

Your sub-theme can be setup to use either CSS or SCSS. SCSS is an extension to CSS and is for more advanced theme developers.

Template



5. Select the Default template to base your **sub-theme** on. You can create a **sub-theme** based on a color template. However, for this scenario you use the default connect\_theme template.
6. Click Generate.
7. Download the generated **sub-theme** to a location of your choice, and extract all the files from the .zip file.

On your computer, navigate to the files that you extracted. Find the overrides.css file in here banka\_theme/banka\_theme/css/overrides.css. You can now customize the overrides.css file.

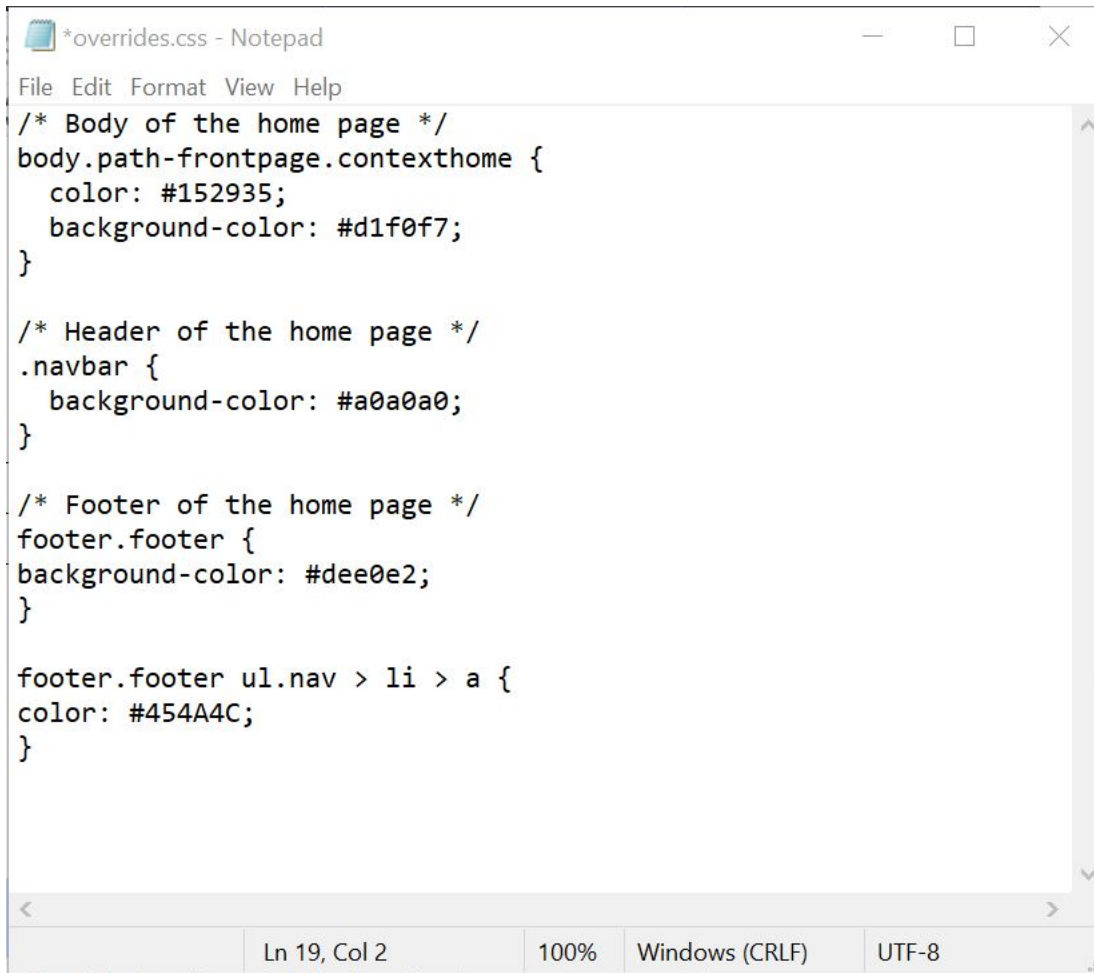
1. Open the overrides.css file in your chosen editor.
2. Customize your **sub-theme** by entering the following elements into the overrides.css file:

```
/* Body of the home page */
body.path-frontpage.contexthome {
 color: #152935;
 background-color: #d1f0f7;
}

/* Header of the home page */
.navbar {
 background-color: #a0a0a0;
}

/* Footer of the home page */
footer.footer {
 background-color: #dee0e2;
}

footer.footer ul.nav > li > a {
 color: #454A4C;
}
```



```
*overrides.css - Notepad
File Edit Format View Help
/* Body of the home page */
body.path-frontpage.contexthome {
 color: #152935;
 background-color: #d1f0f7;
}

/* Header of the home page */
.navbar {
 background-color: #a0a0a0;
}

/* Footer of the home page */
footer.footer {
 background-color: #dee0e2;
}

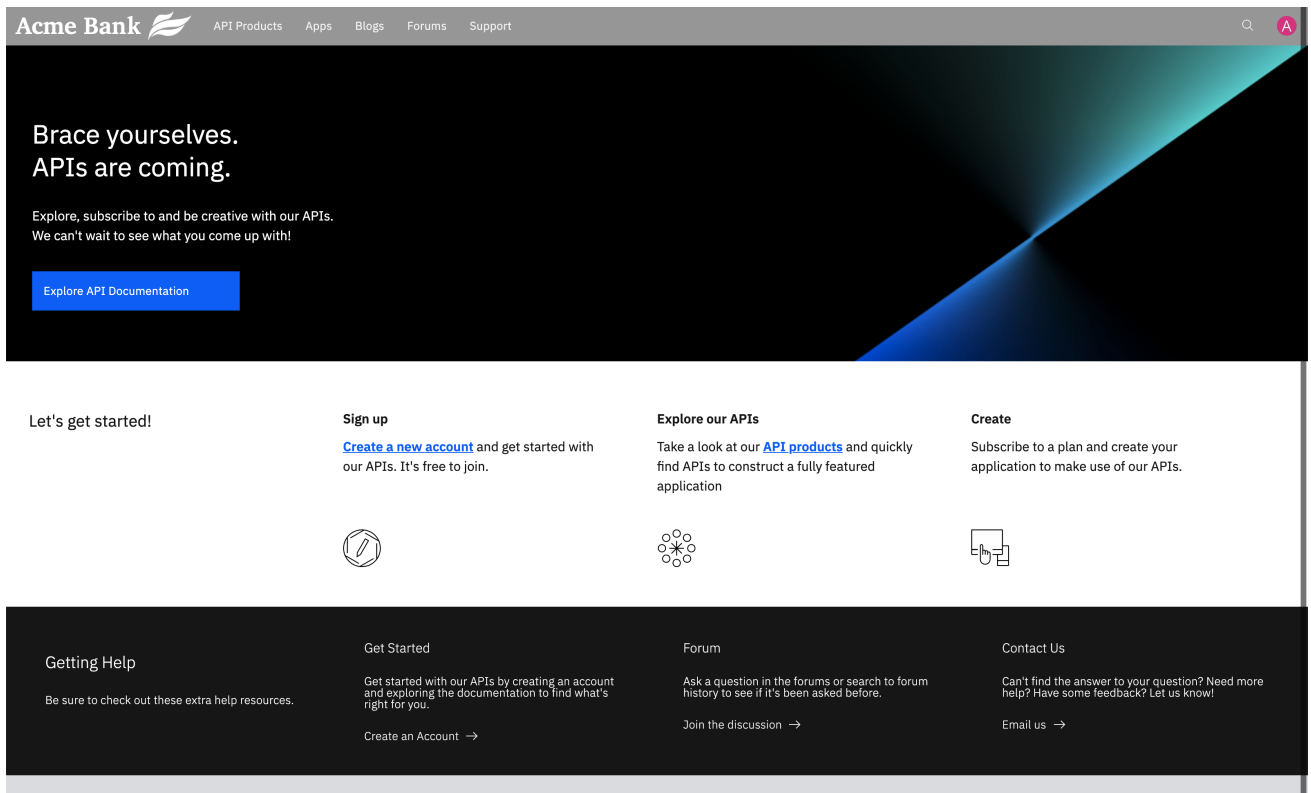
footer.footer ul.nav > li > a {
 color: #454A4C;
}

Ln 19, Col 2 100% Windows (CRLF) UTF-8
```

3. Save the overrides.css file.

You can now install and enable your customized theme.

1. After you finish updating the overrides.css file, compress all the theme files back into the .zip **sub-theme** file that you downloaded originally.
2. In the Developer Portal, click Appearance > Install new theme. The **Install new theme** window is displayed.
3. In Upload a module or theme archive to install click Browse, and navigate to your newly updated compressed theme file.
4. Click Install to install the theme onto your site.
5. Click Enable newly added themes, and find your new theme in the list of **Disabled themes**. Click Enable and set as default to set your new custom **sub-theme** as the default theme for your site.
6. Return to the Developer Portal home page by clicking Back to site. You can now see your custom theme.



## Add a field to an application content type to change the `site-config` on the staging Developer Portal

1. Log in to your staging Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Structure > Content types > Application > Manage fields.
4. Click + Add field.
5. In **Add a new field**, select Text (plain, long). Enter Company name as the label title.

**Add a new field** **Re-use an existing field**  
 or   
**Label \***  
 Machine name: field\_company\_name [Edit]

6. Click Save and continue.
7. Leave the **Allowed number of values** set to 1 and click Save field settings.
8. Click Save field settings. You can now see your new field.



## Use the toolkit CLI to export the custom themes and `site-config` on the staging Developer Portal

1. Log in to the management server.

```
apic login --server management_server --realm realm --username org_username --password org_password
```

For example,

```
apic login -s api.stagingexample.com --realm provider/default-idp-2 --username ibm --password Qwerty123E
```

For more information about variable options, see [Logging in to a management server](#).

2. Export your custom themes.

The **custom-theme:create-export** command creates an export task against the portal of the specified `catalog` and `org`. The command then polls the status of the task until it has **FINISHED** and the artifacts are streamed back.

Note: The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the commands **custom-theme:get-export-status** and **custom-theme:get-export** to get the status and potentially an artifact.

```
apic --mode portaladmin custom-theme:create-export --catalog name/id --org name/id --server management_server
```

- The theme .tgz is saved to the directory in which you ran the command.
- The saved .tgz has the format `custom_theme_createExport-20200217134637.tgz`.
- You can save your exports into your change control management system.

For example,

```
apic --mode portaladmin custom-theme:create-export --catalog dev --org ibm --server api.stagingexample.com
```

### 3. Export your site config.

The `site-config:create-export` command creates an export task against the portal of the specified `catalog` and `org`, the command then polls the status of the task until it has `FINISHED` and the artifacts are streamed back.

```
apic --mode portaladmin site-config:create-export --catalog name/id --org name/id --server management_server
```

- The site config .tgz is saved to the directory in which you ran the command.
- The saved .tgz has the format `site_config_createExport-20200217134637.tgz`.
- You can save your exports into your change control management system.

For example,

```
apic --mode portaladmin site-config:create-export --catalog dev --org ibm --server api.stagingexample.com
```

## Use the toolkit CLI to import the custom themes and `site-config` on the production Developer Portal

### 1. Log in to the management server.

```
apic login --server management_server --realm realm --username org_username --password org_password
```

For example,

```
apic login -s api.productionexample.com --realm provider/default-idp-2 --username ibm --password Qwerty123E
```

For more information about variable options, see [Logging in to a management server](#).

### 2. Import your custom themes.

The `custom-theme:create-import` command uses the supplied .tgz file to import the custom themes into the portal of the specified `catalog` and `org`. The command then polls the status of the task until the task has either `FINISHED` successfully, or failed because of an error.

Note: The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the command `custom-theme:get-import-status` to get the status of the import.

```
apic --mode portaladmin custom-theme:create-import --catalog name/id --org name/id --server management_server /PATH/TO/THEME/TGZ
```

For example,

```
apic --mode portaladmin custom-theme:create-import --catalog prod --org ibm --server api.productionexample.com C:/users/example/desktop/custom_theme_createExport-20200217134637.tgz
```

### 3. Import your site config.

The `site-config:create-import` command uses the supplied .tgz file to import the site config into the portal of the specified `catalog` and `org`. The command then polls the status of the task until the task has either `FINISHED` successfully, or failed because of an error.

```
apic --mode portaladmin site-config:create-import --catalog name/id --org name/id --server management_server /PATH/TO/CONFIG/TGZ
```

For example,

```
apic --mode portaladmin site-config:create-import --catalog prod --org ibm --server api.productionexample.com C:/users/example/desktop/site_config_createExport-20200217134637.tgz
```

## What you did in this scenario

In this scenario, you created a theme and changed the configuration on your staging Developer Portal. Then, you used the toolkit CLI to replicate the changes on to your production Developer Portal.

You can now see the changes that you made on your production Developer Portal.

- [Exporting and importing custom themes and site configuration by using the Portal Admin API](#)

The platform REST APIs for IBM® API Connect provide complete access to the capability of the platform. So, you can use the toolkit CLI to replicate changes from one Developer Portal instance to another. However, if the toolkit CLI was unavailable for some reason, you can use the Portal Admin API.

## Exporting and importing custom themes and site configuration by using the Portal Admin API

The platform REST APIs for IBM® API Connect provide complete access to the capability of the platform. So, you can use the toolkit CLI to replicate changes from one Developer Portal instance to another. However, if the toolkit CLI was unavailable for some reason, you can use the Portal Admin API.

## Before you begin

To complete this scenario, you must have a staging Developer Portal and a production Developer Portal enabled. You must have administrator access, or the permission `settings:manage`, to both to complete this scenario. The tutorial [Creating the Portal](#) explains how to enable the portal if you have not already done so.



Also, you must have downloaded and installed the toolkit. For more information, see [Installing the toolkit](#).

## About this scenario

---

In this scenario, you use the toolkit to obtain some authentication. Then, on your staging Developer Portal, you create a theme and make some configuration changes. Then, you use the Portal Admin API to replicate the changes on your production Developer Portal.

This scenario takes you through the following steps:

1. [Obtaining the client ID and secret, and a bearer token.](#)
2. [Create a custom theme on the staging Developer Portal.](#)
3. [Add a field to an application content type to change the `site-config` on the staging Developer Portal.](#)
4. [Use the Portal Admin API to export the custom themes and `site-config` on the staging Developer Portal.](#)
5. [Use the Portal Admin API to import the custom themes and `site-config` on the production Developer Portal.](#)

## Obtaining the client ID and secret, and a bearer token.

---

From your toolkit:

1. Obtaining the client ID and secret.

- a. Obtain the list of administrator scoped providers to determine the correct realm to use:

```
apic identity-providers:list --scope admin --server <management_server>
```

For example,

```
apic identity-providers:list --scope admin --server api.stagingexample.com
```

Here is an example of some possible output:

```
default-idp-1
```

- b. Run `apic login`, complete the details:

```
apic login --server management_server --realm cloud_realm --username admin_username --password admin_password
```

For example,

```
apic login -s api.stagingexample.com --realm provider/default-idp-1 --username admin --password Qwerty123f
```

- c. Log in to cloud manager, and register an app. Make a new file locally and call it `app1.json`. Then, copy into it the following code snippet:

```
{
 "name": "app1",
 "client_id": "applid",
 "client_secret": "applsecret",
 "client_type": "toolkit"
}
```

- d. Run the registration create command:

```
apic registrations:create --server management_server app1.json
```

For example,

```
apic registrations:create --server api.stagingexample.com app1.json
```

Here is an example of some possible output:

```
app1 [state: enabled] https://api.stagingexample.com/api/cloud/registrations/d2b7572f-3406-42a7-bdcc-85f177860df7
```

2. Now, you can get a bearer token from the provider org.

- a. To obtain a list of provider realms:

```
apic identity-providers:list --scope provider --server management_server
```

For example,

```
apic identity-providers:list --scope provider --server api.stagingexample.com
```

Here is an example of some possible output:

```
default-idp-2
```

- b. Run `apic login`, complete the details:

```
apic login --server management_server --realm provider_realm --username org_username --password org_password
```

For example,

```
apic login -s api.stagingexample.com --realm provider/default-idp-2 --username ibm --password Qwerty123f
```

- c. Create an HTTP request to obtain an access token:

```
curl -k -X POST -d '{"username": "org_username", "password": "org_password", "realm": "provider/default-idp-2", "client_id": "app_id", "client_secret": "app_secret", "grant_type": "password"}' -H 'Content-Type: application/json' -H 'Accept: application/json' https://management_server/api/token
```

For example,



On your computer, navigate to the files that you extracted. Find the overrides.css file in here `banka_theme/banka_theme/css/overrides.css`. You can now customize the overrides.css file.

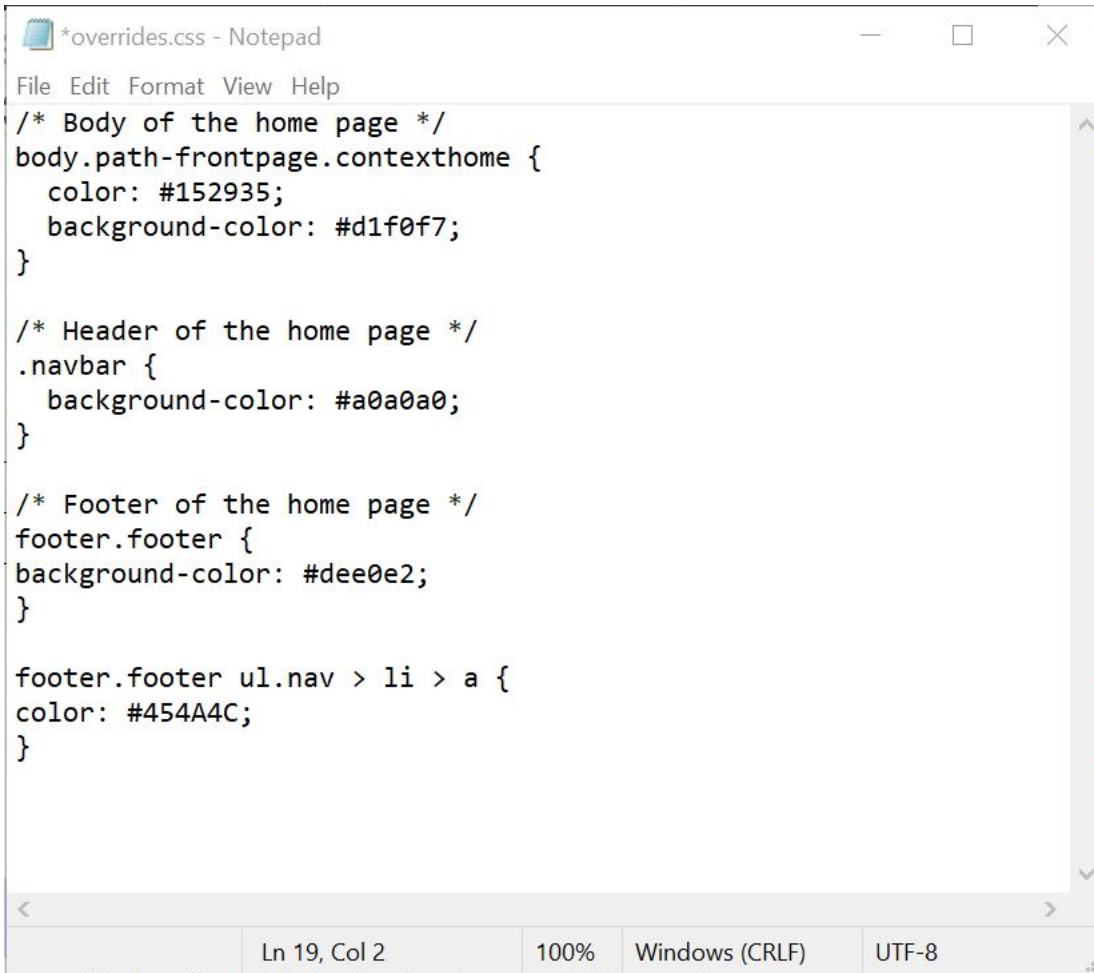
1. Open the overrides.css file in your chosen editor.
2. Customize your **sub-theme** by entering the following elements into the overrides.css file:

```
/* Body of the home page */
body.path-frontpage.contexthome {
 color: #152935;
 background-color: #d1f0f7;
}

/* Header of the home page */
.navbar {
 background-color: #a0a0a0;
}

/* Footer of the home page */
footer.footer {
 background-color: #dee0e2;
}

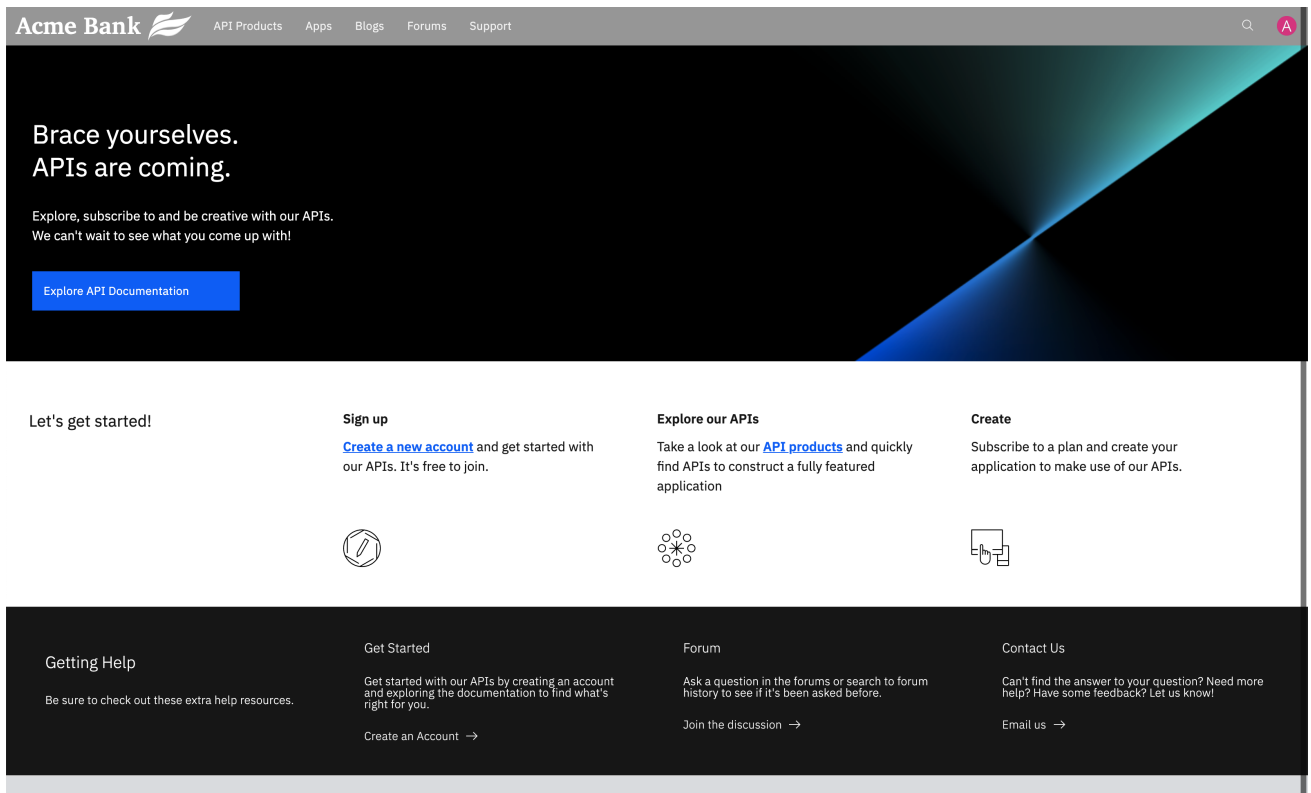
footer.footer ul.nav > li > a {
 color: #454A4C;
}
```



3. Save the overrides.css file.

You can now install and enable your customized theme.

1. After you finish updating the overrides.css file, compress all the theme files back into the .zip **sub-theme** file that you downloaded originally.
2. In the Developer Portal, click Appearance > Install new theme. The **Install new theme** window is displayed.
3. In Upload a module or theme archive to install click Browse, and navigate to your newly updated compressed theme file.
4. Click Install to install the theme onto your site.
5. Click Enable newly added themes, and find your new theme in the list of **Disabled themes**. Click Enable and set as default to set your new custom **sub-theme** as the default theme for your site.
6. Return to the Developer Portal home page by clicking Back to site. You can now see your custom theme.



## Add a field to an application content type to change the `site-config` on the staging Developer Portal

1. Log in to your staging Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Structure > Content types > Application > Manage fields.
4. Click + Add field.
5. In **Add a new field**, select Text (plain, long). Enter Company name as the label title.

**Add a new field**

Text (plain, long) ▼

**Re-use an existing field**

- Select an existing field - ▼

or

**Label \***

Company name | Machine name: field\_company\_name [Edit]

**Save and continue**

6. Click Save and continue.
7. Leave the **Allowed number of values** set to 1 and click Save field settings.
8. Click Save field settings. You can now see your new field.



## Use the Portal Admin API to export the custom themes and `site-config` on the staging Developer Portal

1. Export your custom themes.
  - a. Create the export job by using this command:

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/yaml' 'https://<management_server>/portal/catalogs/<org>/<catalog>/custom-theme/export'
```

For example,

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/yaml' 'https://api.stagingexample.com/portal/catalogs/ibm/staging/custom-theme/export'
```

Here is an example of some possible output:

```
status: 201
message:
 stdout: >-
 /catalogs/75cfdc05-e818-408e-8ce6-fd3fd61c8756/f09602a2-a0e4-40dc-a82d-d1af326ad092/custom-
```

```
theme/export/85z2m39znojolv2s/status
stderr: ''
errors: []
```

- b. You can check the status of the export job by invoking the `/status` endpoint.

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/yaml'
'https://<management_server>/portal/catalogs/<org>/<catalog>/custom-theme/export/<task_id>/status'
```

For example,

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/yaml'
'https://api.stagingexample.com/portal/catalogs/ibm/staging/custom-theme/export/85z2m39znojolv2s/status'
```

Here is an example of some possible output:

```
{
 "status": 200,
 "message": {
 "stdout": "The status of this task is: FINISHED and the artifact can now be streamed"
 },
 "errors": []
}
```

- c. You can now download the custom theme file onto your machine.

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/octet-stream'
'https://<management_server>/portal/catalogs/<org>/<catalog>/custom-theme/export/<task_id>' -o <theme_name>
```

For example,

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/octet-stream'
'https://api.stagingexample.com/portal/catalogs/ibm/staging/custom-theme/export/85z2m39znojolv2s' -o mytheme.tgz
```

Note: When you use the `curl` commands to download files, if you do not specify an output with `-o`, it incorrectly prints the output as binary onto the terminal screen.

- d. On Mac and Linux® machines, you can verify that you have a valid theme export `.tgz` file by invoking the `tar t --list` argument.

```
tar -tvf <theme_name>
```

For example,

```
tar -tvf mytheme.tgz
```

Here is an example of some possible output:

```
drwxrwsr-x 0 aegir aegir 0 9 Jun 13:40 ./
drwxr-sr-x 0 aegir aegir 0 9 Jun 13:40 emerald_theme_css/
-rw-r--r-- 0 aegir aegir 7 9 Jun 13:40 emerald_theme_css/emerald_theme_css.theme
-rw-r--r-- 0 aegir aegir 29224 9 Jun 13:40 emerald_theme_css/screenshot.png
-rw-r--r-- 0 aegir aegir 442 9 Jun 13:40 emerald_theme_css/composer.json
-rw-r--r-- 0 aegir aegir 599 9 Jun 13:40 emerald_theme_css/emerald_theme_css.info.yml
-rw-r--r-- 0 aegir aegir 1150 9 Jun 13:40 emerald_theme_css/favicon.ico
-rw-r--r-- 0 aegir aegir 5932 9 Jun 13:40 emerald_theme_css/logo.svg
-rw-r--r-- 0 aegir aegir 61 9 Jun 13:40 emerald_theme_css/emerald_theme_css.libraries.yml
drwxr-sr-x 0 aegir aegir 0 9 Jun 13:40 emerald_theme_css/css/
-rw-r--r-- 0 aegir aegir 47249 9 Jun 13:40 emerald_theme_css/css/overrides.css
```

## 2. Export your site config.

- a. Create the export job by using this command:

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json'
'https://<management_server>/portal/catalogs/<org>/<catalog>/config/export'
```

For example,

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json'
'https://api.stagingexample.com/portal/catalogs/ibm/staging/config/export'
```

Here is an example of some possible output:

```
{
 "status": 201,
 "message": {
 "stdout": "/catalogs/75cfdc05-e818-408e-8ce6-fd3fd61c8756/f09602a2-a0e4-40dc-a82d-
d1af326ad092/config/export/5cylgk55ciiwn190/status",
 "stderr": ""
 },
 "errors": []
}
```

- b. You can check the status of the export job by invoking the `/status` endpoint.

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json'
'https://<management_server>/portal/catalogs/<org>/<catalog>/config/export/<task_id>/status'
```

For example,

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json'
'https://api.stagingexample.com/portal/catalogs/ibm/staging/config/export/5cylgk55ciiwn190/status'
```

Here is an example of some possible output:

```
{
 "status": 200,
 "message": {
 "stdout": "The status of this task is: FINISHED and the artifact can now be streamed"
 },
 "errors": []
}
```

- c. You can now download the site-config file onto your machine.

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/octet-stream'
'https://<management_server>/portal/catalogs/<org>/<catalog>/config/export/<task_id>' -o <config_name>
```

For example,

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/octet-stream'
'https://api.stagingexample.com/portal/catalogs/ibm/staging/config/export/5cylgk55ciiwn190' -o myconfig.tgz
```

Note: When you use the `curl` commands to download files, if you do not specify an output with `-o`, it incorrectly prints the output as binary onto the terminal screen.

- d. On Mac and Linux machines, you can verify that you have a valid theme export `.tgz` file by invoking the `tar t --list` argument. The `head -n 10` option returns the first 10 lines of the result.

```
tar -tvf <config_name> | head -n 10
```

For example,

```
tar -tvf myconfig.tgz | head -n 10
```

Here is an example of some possible output, which should be a list of yaml files:

```
-rw-rw-r-- 0 aegir aegir 393 11 Jun 15:54 block.block.bartik_page_title.yml
-rw-rw-r-- 0 aegir aegir 577 11 Jun 15:54 filter.format.plain_text.yml
-rw-rw-r-- 0 aegir aegir 256 11 Jun 15:54 captcha.captcha_point.consumerorg_invite_user_form.yml
-rw-rw-r-- 0 aegir aegir 741 11 Jun 15:54 block.block.emerald_theme_css_account_menu.yml
-rw-rw-r-- 0 aegir aegir 488 11 Jun 15:54 field.field.node.application.apic_hostname.yml
-rw-rw-r-- 0 aegir aegir 460 11 Jun 15:54 field.storage.node.apic_catalog_id.yml
-rw-rw-r-- 0 aegir aegir 545 11 Jun 15:54 field.field.user.user.apic_user_registry_url.yml
-rw-rw-r-- 0 aegir aegir 427 11 Jun 15:54 language.content_settings.shortcut.default.yml
-rw-rw-r-- 0 aegir aegir 460 11 Jun 15:54 field.storage.node.api_soapversion.yml
-rw-rw-r-- 0 aegir aegir 148 11 Jun 15:54 captcha.captcha_point.node_product_form.yml
```

## Use the Portal Admin API to import the custom themes and `site-config` on the production Developer Portal

Note: If you are importing your custom theme onto a new server, then before you continue, follow step 1 of this tutorial to obtain a new token to authenticate with your second server. If you are using the same server, and are importing the theme into a new catalog, then the current TOKEN remains valid for the rest of the steps.

Note: You must import the custom theme before you import the site config. This order is because the site config contains information about the default theme, and so your custom theme must exist on the portal when the site config is imported so that it can be set upon import.

1. Import your custom themes.

- a. On the production catalog, import the custom theme onto the production catalog `prod` by using this command:

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/yaml' -H 'Content-Type:
application/octet-stream' 'https://<management_server>/portal/catalogs/<org>/<catalog>/custom-theme/import' --data-
binary @<path_to_theme_file>
```

For example,

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/yaml' -H 'Content-Type:
application/octet-stream' 'https://api.productionexample.com/portal/catalogs/ibm/prod/custom-theme/import' --data-
binary @/Users/steve/PortaladminCLI/mytheme.tgz
```

Here is an example of some possible output:

```
status: 201
message:
 stdout: >-
/catalogs/75cfdc05-e818-408e-8ce6-fd3fd61c8756/56f3d4ca-8be2-48f7-8f91-3d654c2b91f2/custom-
theme/import/vtscpuj08i0hfo6/status
 stderr: ''
errors: []
```

- b. You can check the status of the import job by invoking the `/status` endpoint on this command:

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/yaml'
'https://<management_server>/portal/catalogs/<org>/<catalog>/custom-theme/import/<task_id>/status'
```

For example,

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/yaml'
'https://api.productionexample.com/portal/catalogs/ibm/prod/custom-theme/import/vtscpuj08i0hfo6/status'
```

Here is an example of some possible output:

```
{
 "status": 200,
 "message": {
 "stdout": "2020-06-09 13:00:59: CLI task (custom_theme:import) starting.\n2020-06-09 13:00:59: Importing custom
theme emerald_theme_css\n2020-06-09 13:01:18: CLI task (custom_theme:import) completed successfully.",
 "stderr": ""
 },
}
```

```
 "errors": []
}
```

## 2. Import your site config.

- Import the existing site-config onto the production catalog `prod` by using this command:

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json' -H 'Content-Type: application/octet-stream' 'https://<management_server>/portal/catalogs/<org>/<catalog>/config/import' --data-binary @<path_to_config_import>
```

For example,

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json' -H 'Content-Type: application/octet-stream' 'https://api.productionexample.com/portal/catalogs/ibm/prod/config/import' --data-binary @/Users/steve/PortaladminCLI/myconfig.tgz
```

Here is an example of some possible output:

```
{
 "status": 201,
 "message": {
 "stdout": "/catalogs/75cfdc05-e818-408e-8ce6-fd3fd61c8756/56f3d4ca-8be2-48f7-8f91-3d654c2b91f2/config/import/kzyozx4v5sfa81a5/status",
 "stderr": ""
 },
 "errors": []
}
```

- You can check the status of the config import job by invoking the `/status` endpoint:

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json' 'https://<management_server>/portal/catalogs/<org>/<catalog>/config/import/<task_id>/status'
```

For example,

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json' 'https://api.productionexample.com/portal/catalogs/ibm/prod/config/import/kzyozx4v5sfa81a5/status'
```

Here is an example of some possible output:

```
{
 "status": 200,
 "message": {
 "stdout": "2020-06-11 15:42:55: CLI task (site_config:import) starting.\n2020-06-11 15:43:29: CLI task (site_config:import) completed successfully.",
 "stderr": ""
 },
 "errors": []
}
```

## What you did in this scenario

---

In this scenario, you created a theme and changed the configuration on your staging Developer Portal. Then, you used the Portal Admin API to replicate the changes on to your production Developer Portal.

You can now see the changes that you made on your production Developer Portal.

## How to enable and disable maintenance mode on your Developer Portal

---

You can use the Portal commands in the toolkit CLI to enable and disable maintenance mode on your Developer Portal site.

### Before you begin

---

To complete this Portal CLI scenario, you must have a Developer Portal enabled. You must also have administrator access to this site, and the permission `settings:manage`, to complete this scenario. The tutorial [Creating the Portal](#) explains how to enable the portal if you haven't enabled it already.

For more information about using the toolkit CLI with the Developer Portal, see [Getting started with the Portal CLI commands](#).

Important: Maintenance mode is designed for short-term site maintenance; it is not meant for long-term usage. While a site is in maintenance mode, the database is not updated with new content from API Manager.

### About this scenario

---

In this scenario, you use the toolkit CLI to enable, and then disable, maintenance mode on your Developer Portal site.

The scenario takes you through the following steps:

- [Log in to the management server as a Provider organization by using the toolkit CLI](#)
- [Use the toolkit CLI to enable maintenance mode](#)
- [Use the toolkit CLI to disable maintenance mode](#)

## Log in to the management server as a Provider organization by using the toolkit CLI

---

- Log in to the toolkit CLI as a Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total results: 2
results:
- title: Cloud Manager User Registry
 realm: provider/default-idp-2
- title: Corporate LDAP user registry
 realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

For full details of the `apic login` command, see [Logging in to a management server](#).

## Use the toolkit CLI to enable maintenance mode

---

- Run the following command to enable maintenance mode on your Developer Portal:

```
apic --mode portaladmin drupal-state:set --server management_server --org orgid/name --catalog catalogid/name --state_key system.maintenance_mode --input_format integer --state_value 1
```

Where:

- `management_server` is the endpoint URL of your management server (required).
- `catalogid/name` is the ID or name of the catalog that your site belongs to (required).
- `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
- `--state_key` is the name of the state key (required). Must be set to `system.maintenance_mode`.
- `--input_format` is the input format of the value of the state key. Must be set to `integer` for configuring the maintenance mode.
- `--state_value` is the value to assign to the state key (required). Set to `1` to enable maintenance mode.

For example, to enable maintenance mode for a Developer Portal in the `myOrg` organization, and in the `dev` catalog:

```
apic --mode portaladmin drupal-state:set --server my.management.server.com --org myOrg --catalog dev --state_key system.maintenance_mode --input_format integer --state_value 1
Successfully set state system.maintenance_mode to the value of 1.
```

The Developer Portal site is now inaccessible to everyone apart from an administrator. All other users who enter the site URL will see a maintenance message set by the administrator.

## Use the toolkit CLI to disable maintenance mode

---

- Run the following command to disable maintenance mode on your Developer Portal:

```
apic --mode portaladmin drupal-state:set --server management_server --org orgid/name --catalog catalogid/name --state_key system.maintenance_mode --input_format integer --state_value 0
```

Where:

- `management_server` is the endpoint URL of your management server (required).
- `catalogid/name` is the ID or name of the catalog that your site belongs to (required).
- `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
- `--state_key` is the name of the state key (required). Must be set to `system.maintenance_mode`.
- `--input_format` is the input format of the value of the state key. Must be set to `integer` for configuring the maintenance mode.
- `--state_value` is the value to assign to the state key (required). Set to `0` to disable maintenance mode.

For example, to disable maintenance mode for a Developer Portal in the `myOrg` organization, and in the `dev` catalog:

```
apic --mode portaladmin drupal-state:set --server my.management.server.com --org myOrg --catalog dev --state_key system.maintenance_mode --input_format integer --state_value 0
Successfully set state system.maintenance_mode to the value of 0.
```

Your Developer Portal site is now accessible to everyone again.

## What you did in this scenario

---

In this scenario, you used the toolkit CLI to put your Developer Portal site into maintenance mode, and to turn maintenance mode off again.

## Related tasks

---

- [Toggling the site in and out of maintenance mode](#)

## Troubleshooting guide for the Developer Portal

---


Use this guide to help you diagnose and resolve Developer Portal issues in IBM® API Connect. For example, why is the Developer Portal not connecting to the Management server, why is my site not created, and why can't I log in?



## About

---

The following sections provide advice about how to diagnose and resolve some common problems. This information can help you to identify whether a specific IBM API Connect Version 10 component is failing, whether it's an environmental problem, and whether you should raise a support request. Some advice includes reviewing specific log files so you can narrow down the source of the problem.

Note: In the  Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.

1. [Why is the Developer Portal not connecting to the Management server?](#)
2. [Why is my Developer Portal site not being created?](#)
3. [Why can't I register a user?](#)
4. [Why can't I login?](#)
5. [Why are changes to Products not appearing in the Developer Portal?](#)
6. [Why am I having problems with my Developer Portal user interface?](#)
7. [Why am I having problems with my Kubernetes system?](#)
8. [Why am I having problems with my Developer Portal backup?](#)
9. [Why am I having problems installing Drupal 8 based custom modules or sub-themes into the Drupal 9 based Developer Portal?](#)

## Why is the Developer Portal not connecting to the Management server?

---

Problems with connecting to the Management server can be due to connectivity or TLS issues. You should check the following areas:

- Can the Management server actually reach the Developer Portal?
- Is SSL passthrough enabled or required on any load balancer that sits between the Management server and the Developer Portal? For more information, see [JWT security on Kubernetes and OpenShift](#), [JWT security on OVA](#).

If you need to open a Service Request, include the following information:

- The [IBM API Connect MustGather Information](#).
- Details of the date and time that the connection attempts were made.

## Why is my Developer Portal site not being created?

---

If you are having problems creating a site, you should take the following actions to help diagnose the issue:

1. Tail the nginx container log to confirm that the site create request reached the portal, for example:

```
kubectl logs rb93d4fb118-apic-portal-nginx-7c7b464fb6-vvx5m
```

2. Look in the container log for a site create request, for example:

```
POST /portal-service-configuration-create HTTP/1.1" 200
```

- a. If you don't find a site create request in the log, then the create request is not reaching the Developer Portal from the Management. Follow the instructions in [Why is the Developer Portal not connecting to the Management server?](#) to diagnose the connectivity issues.
- b. If a site create request is in the log, then further diagnostics can be found in the admin pod log, for example:

```
kubectl logs -f rb93d4fb118-apic-portal-www-p4bhj -c admin
```

Look in the admin pod log for messages that refer to `create_site`, as well as the name of the site.

## Why can't I register a user?

---

If you are having problems registering a user, take the following actions to help diagnose the issue:

- The user registration links in the emails are only valid for 24 hours, so check that the system time is correctly configured across all of the pods.
- If emails aren't being received, check that the email server is correctly configured; see [Configuring an email server for notifications](#). Also, if possible, check whether other emails such as publish approval alerts or password reset emails are being received.

If you need to open a Service Request, please take the following steps:

1. Note which user registry type is being used, such as Local User Registry (LUR), or LDAP user registry.
2. Enable the debug REST server and method trace in the Developer Portal UI. Click Configuration > Development > IBM Development on the administrator dashboard. Then select the Enable method entry / exit trace and Enable API Manager REST interface debug check boxes. Click Save configuration.
3. Re-create the issue, making a note of the time, the username, and any messages displayed on the screen.
4. Gather the web container log for the www pod, for example:

```
kubectl logs portal-apic-portal-www-zhqxf web > web.log
```

5. Gather all of the Management server logs and the logs for the user registry type that is being used; see [IBM API Connect MustGather Information](#) for details.

## Why can't I login?

---

If users are having problems with logging in to the Developer Portal, please take the following steps before opening a Service Request:

1. Note which user registry type is being used, such as Local User Registry (LUR), or LDAP user registry.
2. Enable the debug REST server and method trace in the Developer Portal UI. Click Configuration > Development > IBM Development on the administrator dashboard. Then select the Enable method entry / exit trace and Enable API Manager REST interface debug check boxes. Click Save configuration.
3. Recreate the issue, making a note of the time, the username, and any messages printed to the screen.
4. Gather the web container log for the www pod, for example:

```
kubectl logs portal-apic-portal-www-zhqxf web > web.log
```

5. Gather all of the Management server logs and the logs for the user registry type that is being used; see [IBM API Connect MustGather Information](#) for details.

## Why are changes to Products not appearing in the Developer Portal?

To determine whether the problem is with the Portal server (either the nginx or the www pod), the Management server, or the network, please make the following diagnostic checks.

Portal nginx pod

1. Tail the nginx pod log on the Portal, for example:

```
kubectl logs -f portal-apic-portal-nginx-866c977c4c-n9gbf
```

2. Make a publish request to the Catalog (note that the request must be a Publish or Retire operation, not a Stage operation).
3. In the log being tailed, see if the following message appears:

```
...POST /event-create...
```

If the POST message appears in the nginx log, then the publish webhook that was sent from the Management server did arrive at the Portal. If there is an error message in place of a POST message, then the problem is likely to be with the nginx pod. If there is no message at all, then it's likely that the webhook from the Management server is not reaching the Portal and there is a network problem.

Portal www pod

1. Tail the www pod admin container log on the Portal, for example:

```
kubectl logs -f portal-apic-portal-www-dtbzw -c admin
```

2. Make a publish request to the Catalog (note that the request must be a Publish or Retire operation, not a Stage operation).
3. In the log being tailed, see if the following message appears:

```
Received webhook event 'product_lifecycle' for catalog
```

If no such message appears, it suggests that the webhook is either not arriving at the Portal, or not getting beyond the nginx pod (see the previous section [Portal nginx pod](#)). Any errors that the admin container has in processing the webhook will be logged in the www pod admin container log.

Management server apim pod

1. Tail the apim pod log on the Management server, for example:

```
kubectl logs -f apiconnect-apim-v2-86b464c4f5-4cx6r
```

2. Make a publish request to the Catalog (note that the request must be a Publish or Retire operation, not a Stage operation).
3. In the log being tailed, see if there any error messages.

Search for the word 'sending', you should see the message `webhook: sending successful` if the webhook is sent, or error messages indicating why if the webhook is not sent.

## Why am I having problems with my Developer Portal user interface?

If you are having user interface (UI) problems, make the following checks:

- Try clearing the cache from your browser, try using private browsing, and try using different browsers - see whether there are any differences in behavior.
- Check your browser window size and screen resolution, and try increasing and decreasing the settings.
- Investigate the browser developer tools console and network tabs, as they might provide some diagnostic clues.
- Try clearing the Drupal caches from within the Developer Portal; see [Clearing the server caches](#) for details.

If you need to open a Service Request, along with the [IBM API Connect MustGather Information](#) logs, also include details of the time and date when the problem was reproduced. In addition, an export of the browser developer tools console and network output would be helpful.

## Why am I having problems with my Kubernetes system?

Kubernetes provides a DNS server as one of the pods in the kube-system namespace (unless you have configured your Kubernetes setup differently), and sometimes this DNS server can silently fail. When the Portal www or db pods restart, they check that their own hostname and IP address is in the service that is defined for the pod. If the DNS server is down, this check fails. An indication that the DNS server is down is seeing the following log lines printed endlessly in the admin or db containers:

```
[admin stdout] 2cd141dc:2cd141dc:2cd141dc 2018-10-08 07:27:54: service-ready:
[admin stdout] 2cd141dc:2cd141dc:2cd141dc 2018-10-08 07:27:54: service-all:
[admin stdout] 2cd141dc:2cd141dc:2cd141dc 2018-10-08 07:27:54: Waiting for sleep in service all check. Pid(s) 352 Seconds
24
[admin stdout] 2cd141dc:2cd141dc:2cd141dc 2018-10-08 07:27:59: Finished waiting for sleep in service all check. Pid(s)
352 Seconds 29
[admin stdout] 2cd141dc:2cd141dc:2cd141dc 2018-10-08 07:28:01: WARNING 5: r4c09b98d02-apic-portal-admin-all doesn't
include this pod or r4c09b98d02-apic-portal-director does. Checking again in 5 seconds.
```

To fix this problem, delete the DNS pod in the kube-system namespace to force it to restart, for example:

```
kubectl -n kube-system delete pod kube-dns-b76db4f7f-n4lvt
```

Where `kube-dns-b76db4f7f-n4lvt` is the full name of the DNS pod.

Note: To find out the full name of the DNS pod in your Kubernetes system, run the following command:

```
kubectl -n kube-system get pods
```

This command will return data like the following example:

```
$ kubectl -n kube-system get pods
NAME READY STATUS RESTARTS AGE
default-http-backend-5bccfbdc8-mjnnb 1/1 Running 0 4d
etcd-minikube 1/1 Running 0 4d
kube-addon-manager-minikube 1/1 Running 0 4d
kube-apiserver-minikube 1/1 Running 0 4d
kube-controller-manager-minikube 1/1 Running 0 4d
kube-dns-6f4fd4bdf-7zslq 3/3 Running 0 4d
kube-proxy-mddmj 1/1 Running 0 4d
kube-registry-proxy-99qml 1/1 Running 0 4d
kube-registry-v0-zt8pv 1/1 Running 0 4d
kube-scheduler-minikube 1/1 Running 0 4d
kubernetes-dashboard-77d8b98585-dczcs 1/1 Running 0 4d
nginx-ingress-controller-57bcfc76d6-z775d 1/1 Running 0 4d
storage-provisioner 1/1 Running 0 4d
tiller-deploy-587df449fb-bpwhq 1/1 Running 0 4d
```

## Why am I having problems with my Developer Portal backup?

If you attempt to back up your portal system or site:

```
apicup subsys exec <portal_subsystem> backup-system|backup-site|backup-all
```

You might encounter the following message:

```
curl: (6) Could not resolve host: apic-portal-apic-portal-director
```

This error is the result of a failure of the SFTP backup file transfer to the backup system, caused by missing authentication with the portal node. To correct this error, `ssh` from the backup system to the portal node, and accept the authentication key when prompted:

1. Enter the portal-www admin container, for example:

```
kubectl exec -it WWW_POD_NAME -c admin bash
```

2. `ssh` to your backup server and accept any authentication prompts, for example:

```
ssh BACKUP_SERVER_HOSTNAME
```

When you have successfully authenticated with your backup server, you can close the session. This action needs to be done only once. Portal backups can then run without any issues.

## Why am I having problems installing Drupal 8 based custom modules or sub-themes into the Drupal 9 based Developer Portal?

From IBM API Connect 10.0.3.0, the Developer Portal is based on the Drupal 9 content management system. If you want to install Drupal 8 custom modules or sub-themes into the Drupal 9 based Developer Portal, you must ensure that they are compatible with Drupal 9, including any custom code that they contain, and not using any deprecated APIs, for example. There are tools available for checking your custom code, such as [drupal\\_check](#) on GitHub, which checks Drupal code for deprecations.

For example, any Developer Portal sites that contain modules or sub-themes that don't contain a Drupal 9 version declaration will fail to upgrade, and errors like the following output will be seen in the `admin` logs:

```
[queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: Checking theme: emeraldgreen
[queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '' found for emeraldgreen
[queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: Checking theme: rubyred
[queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '8.x' found for rubyred
[queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: Found themes incompatible with
Drupal 9: emeraldgreen rubyred
[queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: /tmp/restore_site.355ec8 is NOT
Drupal 9 compatible
...
[queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: Checking module: custom_mod_1
[queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '' found for custom_mod_1
[queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: Checking module: custom_mod_2
[queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '8.x' found for custom_mod_2
[queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: Found modules incompatible with
Drupal 9: emeraldgreen rubyred
[queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: sitel.com is NOT Drupal 9
compatible
```

To fix version compatibility errors, all custom modules and sub-themes should declare a `core_version_requirement` key in their `*.info.yml` file that indicates Drupal 9 compatibility. For example:

```
name: Example module
type: module
description: Purely an example
core: 8.x
core_version_requirement: '^8 || ^9'
package: Example module

Information added by Drupal.org packaging script on 2020-05-31
version: '8.x-1.3'
project: 'example_module'
datestamp: 1590905415
```

This example specifies that the module is compatible with all versions of Drupal 8 and 9. For more information, see [Let Drupal know about your module with an .info.yml file](#) on the drupal.org website.

If you have a backup of a site that you need to restore and are getting the version compatibility error, but the module or theme \*.info.yml file cannot be changed easily, then you can modify the site backup.

To modify the site backup, extract the it, edit the relevant files inside it, and then tar the backup file again. Note that this procedure will overwrite the original backup file, so ensure that you keep a separate copy of the original file before you start the extraction. For example:

```
1. mkdir /tmp/backup
2. cd /tmp/backup
3. tar xzf path_to_backup.tar.gz
4. Edit the custom module and theme files to make them Drupal 9 compatible, and add the correct core_version_requirement setting.
5. rm -f path_to_backup.tar.gz
6. tar cfz path_to_backup.tar.gz
7. cd /
8. rm -rf /tmp/backup
```

## Related information

---

- [IBM Support Service Requests and PMRs](#)

---

## Migrating your Developer Portal from Version 5 to Version 10

Guidance on how to migrate a Developer Portal site from IBM® API Connect Version 5, to IBM API Connect Version 10.

In IBM API Connect Version 10 the Developer Portal is based on the open source Drupal 9 content management software, which enables more advanced, flexible, and powerful customization capabilities. However, in IBM API Connect Version 5 the Developer Portal is based on Drupal 7. The migration tooling in Version 10 will create a new Developer Portal site, and migrate all of the associated APIs and Products, user information, and subscriptions over. However, customizations such as modules and themes cannot be automatically migrated, as the Drupal 9 baseline is very different to Drupal 7. For more information about upgrading from a Drupal 7 baseline to a Drupal 9 baseline, see [Upgrading Drupal](#) in the Drupal documentation.

The following sections take you through the migration process for the Developer Portal, and provide guidance on how to manually migrate any specific customizations to Drupal 9.

- [Migration process for the Developer Portal](#)
- [Portal Delegated User Registries](#)
- [Custom themes](#)
- [Custom modules](#)
- [Custom Rules](#)
- [Custom content types and fields](#)
- [Advanced theme development](#)
- [Changing the page layout](#)
- [Changing the front page](#)
- [Features no longer available in the Developer Portal](#)

Note: The ability to migrate is available only from IBM API Connect Version 5.0.8.7 onwards. If your deployment is at an earlier version, you must upgrade before migrating. See [Upgrading your API Connect cloud](#) for information about upgrading in Version 5. For more information about the supported versions for migration, see [Migrating a Version 5 deployment](#).

Note: In Version 5, it was possible, though not recommended, to modify files on the Developer Portal server directly. For more information, see [Developer Portal best practices for administrators](#). However, in Version 10, you should not make any system changes. Do not use the `kubect exec` command to make changes inside the Developer Portal containers, and on OVA deployments do not make any modifications on the Developer Portal VM. Make any customizations on your Developer Portal by using the UI or CLI. Do not run a `kubect exec` command on any Developer Portal containers unless asked to by IBM.

---

## Migration process for the Developer Portal

The migration process for the Developer Portal migrates all of the following information to Version 10 for each site in your cluster:

- User information (if Portal Delegated User Registry is enabled, the process is slightly different; see [Portal Delegated User Registries](#)).
- Products and APIs.
- Consumer organizations.
- Applications.
- Subscriptions.

For information about the main migration process, see [Migrating a Version 5 deployment](#).

---

## Portal Delegated User Registries

In IBM API Connect Version 5, if Portal Delegated User Registry (PDUR) was selected for a Catalog, the user management was delegated from the management server to the Developer Portal. The site administrator within the Portal then performed the configuration of user registries, and the management server recorded that the delegation had taken place, and stored a record of the users that existed in the Catalog. However, the management service maintained no information about the configuration of the specific registries or how users mapped to them. In IBM API Connect Version 10, all user registry configuration takes place in the management server, and PDUR is no longer needed. Therefore, to migrate the PDUR user registries and users from v5 to v10, configuration information must be gathered from both the management server and the Portal node.

If your IBM API Connect Version 5 Developer Portal uses Portal Delegated User Registry (PDUR), you must first export the user information that is stored in the Version 5 Developer Portal local database. This user information can then be fed into the migration tooling so that the correct user registries can be created in the newer version of API Connect.

For more information about how to export the PDUR information from v5, see [Exporting Portal Delegated User Registry user information](#). For more information about the migration process in v10, see [Migration steps with PDUR](#).

---

## Custom themes

If your Version 5 Developer Portal is using a custom theme, you'll need to create the custom theme again for the Drupal 9 baseline. Start by creating a sub-theme of the new Version 10 Developer Portal site by using the theme generator, and then manually customize the sub-theme to your specifications for Drupal 9 by using Cascading Style Sheets (CSS), or Sass CSS (SCSS). Note that using SCSS in your theme makes it far simpler to change colors. We've also added a color template theme generator, which means you can quickly generate a base color theme on which to build your customized site branding. For more information, see [Creating a sub-theme](#).

Note, directly editing the API Connect theme is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed.

Other useful resources:

- [Tutorial: Creating a custom theme for the Developer Portal](#) - takes you through how to use the theme generator, customize a theme, and install a custom theme.
- [Changing the site logo](#) - how to change your site logo.
- [Changing the shortcut icon](#) - how to change the shortcut icon (favicon).
- [Creating sub-themes](#) - Drupal documentation about creating a sub-themes.
- [Theming differences between Drupal 6, 7 & 8](#) - Drupal documentation about theming differences.

---

## Custom modules

If your Version 5 Developer Portal is using any custom modules, you will need to create these custom modules again for the Drupal 9 baseline. For more information on writing custom modules, see [Extend](#).

Other useful resources:

- [Creating custom modules](#) - Drupal documentation about creating custom modules.
- [Getting started creating custom modules](#) - Drupal documentation giving background information and prerequisites needed for custom module development.
- [Understanding hooks](#) - Drupal documentations about hooks.
- [Drupal API hooks](#) - Drupal documentation about the hooks that are available.

---

## Custom Rules

If your Version 5 Developer Portal is using any custom Rules, you will need to create them as custom modules for the Drupal 9 baseline. You cannot export the Drupal 7 Rules, and then import them into the Drupal 9 baseline, as the events and actions are implemented in a different way. For more information about custom modules, see the [Custom modules](#) section.

---

## Custom content types and fields

If your Version 5 Developer Portal is using any custom content, you will need to create this content again for the Drupal 9 baseline.

For information about creating custom content types, see the following topics:

- [Creating content types](#)
- [Adding fields to content types](#)
- [Adding custom fields to user records](#)

---

## Advanced theme development

If your Version 5 Developer Portal is using modified content type templates, you must create these templates again for the Drupal 9 baseline. In Drupal 9 the templates are Twig files that define the actual HTML output for a particular piece of content. You can override a Twig template by copying the original template into the templates folder of your custom sub-theme, and then editing the template to your specification. The Developer Portal then uses your template in preference to the original one. For more information, see [Applying a modified content type template](#), and [Creating a sub-theme](#). You can also follow a guided example of how to create a new theme, see [Tutorial: Creating a custom theme for the Developer Portal](#).

Note:

Modifying Twig templates allows very fine-grained control over the structure of pages. However, it also means you might miss new features and defect fixes that are made to the original templates, as your Developer Portal is using your overrides instead of the originals. So, if you override a template, it is your responsibility to check the templates in the latest API Connect Version 10 releases, and to ensure that any equivalent changes that are needed to your overrides to maintain functional equivalence are made.

---

## Changing the page layout

If your Version 5 Developer Portal is using a different page layout, you will need to create this layout again for the Drupal 9 baseline. The page layout is defined by configuring which blocks appear in which regions. Blocks are controlled by using the Structure\_>Block layout options on the Developer Portal administrator dashboard.

For information about how to configure the page layout, see [Adding and changing the blocks displayed on Developer Portal pages](#).

---

## Changing the front page

If your Version 5 Developer Portal is using a different front page, you will need to create this front page again for the Drupal 9 baseline. The front page is configured from the Structure\_>Pages options on the Developer Portal administrator dashboard.

For information about how to configure the front page, see [Configuring the front page](#), and [Adding custom pages](#).

Further options that you might want to consider when designing your front page include:

- Change the banner block content or image: [Changing the front page banner block](#).
- Display featured API Products by creating a featured content block: [Changing the front page Featured Content block](#).
- Create custom blocks with your own HTML content: [Adding and changing the blocks displayed on Developer Portal pages](#).
- Set up the social block to include your organization Twitter feed: [Integrating Twitter data into the social block](#).

---

## Features no longer available in the Developer Portal

---

The following features that are available in IBM API Connect Version 5, are not available in IBM API Connect Version 10:

- Security questions - these can now be created inside an OpenID Connect provider.
- Two-factor authentication - this can now be performed inside an OpenID Connect provider.
- Support tickets - no Drupal 9 equivalent module is currently available.

---

## Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10

---

Information about the upgrade of the Developer Portal from Drupal 9 to Drupal 10.

The Developer Portal is based on the open source Drupal content management system. From API Connect 10.0.5.3, Drupal 9 is upgrading to Drupal 10, which also requires PHP 8.1. All existing Developer Portal sites will be upgraded to, and all new sites will be based on Drupal 10. Therefore, all modules and themes on your Developer Portal site must be compatible with Drupal 10 and PHP 8.1. The core modules and themes are automatically made compatible as part of the upgrade. However, if your Developer Portal sites contain any custom modules or themes, either written in-house or contributed by the Drupal community, it is your responsibility to ensure that they are Drupal 10 and PHP 8.1 compatible before the upgrade.

See the following topics for more detailed information about how to prepare your Developer Portal sites for the upgrade to Drupal 10.

- [How to find your custom modules and themes on your Developer Portal](#).
- [Prepare your Developer Portal for the update for Drupal 10](#).
- [Upgrade your custom modules to be Drupal 10 and PHP 8.1 compliant](#).

---

## Upgrading from older versions of API Connect v10

---

If you are upgrading to 10.0.5.3 from 10.0.1.7 or earlier, or coming from 10.0.5.0 or earlier, your Developer Portal sites will first be upgraded to Drupal 9.5 and then to Drupal 10. This is because upgrading to Drupal 10 requires your site to be on at least Drupal 9.4.4.

Note that if you are in the category where a double upgrade occurs, your Developer Portal sites are going to take double the time to upgrade compared to your normal experience, because upgrading from Drupal 9 to Drupal 10 is a major upgrade. Therefore, when you are planning your maintenance window for upgrading to 10.0.5.3, you need to be aware of the extended time frame. To shorten your upgrade window, consider removing the Developer Portal sites that you do not need.

---

## Further reading

---

See the following links for additional information about the Developer Portal upgrade to Drupal 10.

- [Deprecation checking and correction tools](#).
- [Deprecated modules and themes](#).
- [List of APIs deprecated and now removed from Drupal 10](#).
- [Whats new in PHP 8.1](#).
- [Symfony Upgrade Guide](#).
- [Drush 11 commands](#).
- [How to find your custom modules and themes on your Developer Portal](#)  
Guidance on how to find what custom modules and themes you installed on your Developer Portal.
- [Prepare your Developer Portal for the update for Drupal 10](#)  
Guidance on how to prepare your Developer Portal site from Drupal 9 to Drupal 10.
- [Upgrade your custom modules to be Drupal 10 and PHP 8.1 compliant](#)  
Guidance on how to upgrade your custom modules to be Drupal 10 and PHP 8.1 compliant.

---

## How to find your custom modules and themes on your Developer Portal

---

Guidance on how to find what custom modules and themes you installed on your Developer Portal.

The Developer Portal within API Connect comes with built-in modules and themes that you can use to customize your site. You can extend your customizations by installing external custom modules and themes, or even ones written by yourself. If you are unsure which custom modules and themes are installed on your Developer Portal, you can use one of the following methods to find out.

1. [Method 1. Developer Portal admin status report \(recommended\)](#)
2. [Method 2. Developer Portal toolkit CLI](#)
3. [Method 3. Developer Portal container](#)

---








## Method 1. Developer Portal admin status report (recommended)

---

1. Log in to your Developer Portal as the administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Manage > Reports > Status report.

- The custom modules and themes are listed in the Custom Modules and Custom Themes sections at the end of the page.

## General System Information

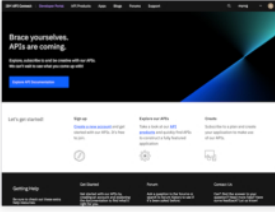

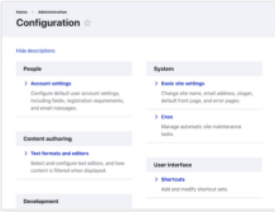

 <b>IBM API Developer Portal Version</b> 10.99.99.99 20230310-1122 <b>API Explorer</b> 6.3.418 Thu Feb 09 2023 10:51:05 GMT+0000 (Greenwich Mean Time) <b>Consumer Analytics</b> 0.1.33 Fri Nov 18 2022 14:20:50 GMT+0000 (Coordinated Universal Time)	 <b>Drupal Version</b> 10.0.4	 <b>Last Cron Run</b> Last run 2 hours 35 minutes ago <a href="#">(more information)</a> <a href="#">Run cron</a>
 <b>Web Server</b> nginx/1.22.1	 <b>PHP</b> <b>Version</b> 8.1.16 <a href="#">(more information)</a> <b>Memory limit</b> 768M	 <b>Database</b> <b>Version</b> 8.0.30-22.1 <b>System</b> MySQL, Percona Server, or equivalent
 <b>Custom Modules</b> chosen google_tag snowflakes	 <b>Custom Themes</b> emerald_theme_css ruby_theme_scss sapphire_theme_css	 <b>Custom User Fields</b> None installed

4. Check which of your custom modules are enabled or disabled by clicking **Manage > Extend**.
  - The enabled modules are displayed with a selected checkbox, and the disabled ones are not selected as seen in the following example.

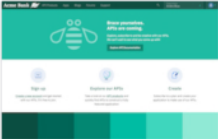



<input checked="" type="checkbox"/>	<b>Page Load Progress</b>	Locks the screen and adds a progress icon to indicate progress when a page takes a long time to reload.
<input type="checkbox"/>	<b>Snowflakes</b>	Adds pure CSS animated snowflakes to your Drupal site.

5. Check which of your custom themes are enabled or disabled by clicking **Manage > Appearance**.
  - Find your themes that are categorized into Enabled and Disabled sections.

### Enabled themes

	<b>connect_theme 8.x-99.0.15 (default theme)</b> Uses the Bootstrap framework Sass source files and must be compiled.		<b>Bootstrap 8.x-3.27</b> Built to use Bootstrap, a sleek, intuitive, and powerful front-end framework for faster and easier web development.
	<b>Claro 10.0.4 (administration theme)</b> A clean, accessible, and powerful Drupal administration theme.		<b>Olivero 10.0.4</b> A clean, accessible, and flexible Drupal front-end theme.

### Disabled themes

			
-------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------

## Method 2. Developer Portal toolkit CLI

Use the `custom-module:create-export` and `custom-theme:create-export` commands to extract your custom modules and themes into a compressed file.

The compressed file includes a list of your custom modules and themes. For more information about using the Developer Portal toolkit CLI and all the other features it offers, see [Overview of the Developer Portal command-line tool](#).

1. To list your modules, implement the following steps:
  - a. Log in to the Developer Portal toolkit CLI as a provider organization.
  - b. Run the following `custom-module:create-export` command against the Developer Portal.

```
apic custom-module:create-export --mode portaladmin [flags]
```

- c. Specify the appropriate `org` and `catalog` name.

For example,

```
> SERVER=myserver.ibm.com
> apic login -s $SERVER -r provider/default-idp-2 -u steve
```

```
Warning: Using default toolkit credentials.
Enter your API Connect credentials
Password?
Logged into myserver.ibm.com successfully
```

```
> apic -s $SERVER -m portaladmin custom-module:create-export -o ibm -c api-connect-catalog-1
201 CREATED - Task ID: x8jupowmktkf8f2r
```

```
Response Code - 202:
```

```
Message(s) -
The status of this task is: QUEUED
```

```
Response Code - 200:
```

```
Message(s) -
The status of this task is: FINISHED and the artifact can now be streamed
```

```
Saving File (Large files may take a while)...
Incoming project saved to custom_module_createExport-20230322143819.tgz
```

After the command has run, extract your compressed export to see the modules inside. Or use `tar` directly as shown in the following example.

```
> tar --exclude='*/*' -tvf custom_module_createExport-20230322143819.tgz
drwxr-sr-x aegir/aegir 0 2023-03-22 12:05 chosen/
drwxr-sr-x aegir/aegir 0 2023-03-22 12:05 google_tag/
drwxr-sr-x aegir/aegir 0 2023-03-22 12:05 snowflakes/
```

2. To check which one of your custom modules are enabled or disabled, implement the following steps:
  - a. Run the following `modules:list` command to print the list of installed modules.

```
apic modules:list --mode portaladmin [flags]
```

- b. Search for your custom modules from the result.  
For example,

```
> SERVER=myserver.ibm.com
> apic -s $SERVER -m portaladmin modules:list -o ibm -c api-connect-catalog-1 --no-core
.
.
.
chosen:
 package: 'User interface'
 display_name: 'Chosen (chosen)'
 status: Disabled
 version: 3.0.5
google_tag:
 package: Statistics
 display_name: 'Google Tag Manager (google_tag)'
 status: Enabled
 version: 8.x-1.6
snowflakes:
 package: 'User interface'
 display_name: 'Snowflakes (snowflakes)'
 status: Enabled
 version: 2.0.1
.
.
```

3. To list your custom themes, implement the following steps:
  - a. Log in to the Developer Portal toolkit CLI as a provider organization.
  - b. Run the following `custom-theme:create-export` command against the Developer Portal.

```
apic custom-theme:create-export --mode portaladmin [flags]
```

- c. Specify the appropriate `org` and `catalog` name.  
For example,

```
> SERVER=myserver.ibm.com
> apic login -s $SERVER -r provider/default-idp-2 -u steve
Warning: Using default toolkit credentials.
Enter your API Connect credentials
Password?
Logged into api.fyre-ci-136264-master.fyre.ibm.com successfully
```

```
> apic -s $SERVER -m portaladmin custom-theme:create-export -o ibm -c api-connect-catalog-1
201 CREATED - Task ID: oh595vuywxsz3526
```

```
Response Code - 202:
```

```
Message(s) -
The status of this task is: QUEUED
```

```
Response Code - 200:
```

```
Message(s) -
The status of this task is: FINISHED and the artifact can now be streamed
```

```
Saving File (Large files may take a while)...
Incoming project saved to custom_theme_createExport-20230322143856.tgz
```

After the command has run, extract your compressed export to see the themes inside. Or use `tar` to view the contents directly as shown in the following example.



```

> tar --exclude='*/' -tvf custom_theme_createExport-20230322143856.tgz
drwxr-sr-x aegir/aegir 0 2023-03-22 14:36 emerald_theme_css/
drwxr-sr-x aegir/aegir 0 2023-03-22 14:36 ruby_theme_scss/
drwxr-sr-x aegir/aegir 0 2023-03-22 14:36 sapphire_theme_css/

```

4. To check which of your custom themes are enabled or disabled, implement the following steps:
  - a. Run the following `themes:list` command to print the list of installed themes.

```
apic themes:list --mode portaladmin [flags]
```

- b. Search for your custom themes from the result.  
For example,

```

> SERVER=myserver.ibm.com
> apic -s $SERVER -m portaladmin themes:list -o ibm -c api-connect-catalog-1 --no-core
bootstrap:
 package: Bootstrap
 display_name: 'Bootstrap (bootstrap)'
 status: Enabled
 version: 8.x-3.27
.
.
.
emerald_theme_css:
 package: Bootstrap
 display_name: 'emerald_theme_css (emerald_theme_css)'
 status: Disabled
 version: null
ruby_theme_scss:
 package: Bootstrap
 display_name: 'ruby_theme_scss (ruby_theme_scss)'
 status: Disabled
 version: null
sapphire_theme_css:
 package: Bootstrap
 display_name: 'sapphire_theme_css (sapphire_theme_css)'
 status: Disabled
 version: null

```

## Method 3. Developer Portal container

If you are unable to access the Developer Portal settings or the Developer Portal toolkit CLI, check the Developer Portal container running within your Kubernetes or Red Hat OpenShift environment.

Note: You need access to the Developer Portal Kubernetes pods to be able to perform these steps.

1. Use `kubectl` or `oc` for Red Hat OpenShift environment to obtain a shell to the Developer Portal running container.

Your Developer Portal sites are located within the `portal-www` pod in the `admin` container within the namespace that your Developer Portal subsystem is deployed to as seen in the following example.

```

> kubectl get pods -n apic | grep www
portal-014179b4-www-0 2/2 Running 0 8d

> kubectl exec -n apic -it portal-014179b4-www-0 -c admin -- bash
bash-4.4$

```

2. After you access the container, use the command `list_sites -a` to obtain the site alias for your Developer Portal. For example,

```

bash-4.4$ list_sites -a
daae4642-b1e2-4fd8-a5a7-77a68ed812ef.944dce4f-bbad-41ce-b166-aa2c9f8c4b17 => ibm.api-connect-catalog-1.portal.fyre-ci-136264-master.fyre.ibm.com (INSTALLED)
daae4642-b1e2-4fd8-a5a7-77a68ed812ef.c560c004-852d-42c3-ae4e-797144a857af => ibm.api-connect-catalog-2.portal.fyre-ci-136264-master.fyre.ibm.com (INSTALLED)

```

The Developer Portal `portal.fyre-ci-136264-master.fyre.ibm.com/ibm/api-connect-catalog-1` has the site alias `ibm.api-connect-catalog-1.portal.fyre-ci-136264-master.fyre.ibm.com`.

3. After you identify the Developer Portal that you want to look at, use the alias string within a `find` command to list your modules and themes, as shown in the following example.

```

bash-4.4$ find /web/platforms/devportal*/sites/myserver.ibm.com/modules -mindepth 1 -maxdepth 1 -type d -exec basename {} \;
\;
chosen
google_tag
snowflake

bash-4.4$ find /web/platforms/devportal*/sites/ibm.api-connect-catalog-1.portal.fyre-ci-136264-master.fyre.ibm.com/themes -mindepth 1 -maxdepth 1 -type d -exec basename {} \;
emerald_theme_css
ruby_theme_scss
sapphire_theme_css

```

4. To check which of your custom modules and themes are enabled or disabled, implement the following sub-steps:
        - a. Use the `drush` command-line tool within the container to print the list of installed modules and themes for your Developer Portal.
        - b. Then, use the site alias that you obtained earlier to run the following `drush` command and find your custom modules and themes.

```

bash-4.4$ drush @myserver.ibm.com pml --no-core
Running: drush -y @daae4642-b1e2-4fd8-a5a7-77a68ed812ef-944dce4f-bbad-41ce-b166-aa2c9f8c4b17 pml --no-core

```

```

Package Name

```

Status	Version	
User		Account Field Split (account_field_split)
Enabled		
Access control		ACL (acl)
Enabled		
.		
.		
User interface		Chosen (chosen)
Disabled	3.0.5	
Statistics		Google Tag Manager (google_tag)
Disabled	8.x-1.6	
User interface		Snowflakes (snowflakes)
Disabled	2.0.1	
Bootstrap		Bootstrap (bootstrap)
Enabled	8.x-3.27	
Bootstrap		connect_theme (connect_theme)
Enabled	8.x-99.0.15	
Bootstrap		emerald_theme_css (emerald_theme_css)
Disabled		
Bootstrap		ruby_theme_scss (ruby_theme_scss)
Disabled		
Bootstrap		sapphire_theme_css (sapphire_theme_css)
Disabled		

## Results

By using one of the three methods discussed in the topic, you managed to successfully list the custom modules and themes installed on your Developer Portal. This also allowed you to see which of these custom modules and themes are enabled or disabled.

1. [Method 1. Developer Portal admin status report \(recommended\)](#)
2. [Method 2. Developer Portal toolkit CLI](#)
3. [Method 3. Developer Portal container](#)

## Next steps

Ensure that you keep your custom modules and themes up to date, and remove deprecated code within your own written custom modules and themes. If you are not using a custom module or theme, consider removing it. This helps mitigate problems when you want to migrate or upgrade your Developer Portal.

## Prepare your Developer Portal for the update for Drupal 10

Guidance on how to prepare your Developer Portal site from Drupal 9 to Drupal 10.

If your Developer Portal contains custom modules and themes that are written by yourself or downloaded from elsewhere, ensure that they are Drupal 10 compatible before you start the upgrade process. This action aims toward a seamless IBM® API Connect upgrade experience. To find more information about your modules and themes compatibility, download the Upgrade Status module available [here](#).

Note: Before you start the upgrade, check the status of your modules and themes and if they are in use. Remove the ones that you do not use. To learn more on how to prepare your Developer Portal for the Drupal 10 upgrade, check the following steps.

1. [Installing the Upgrade Status module](#)
2. [Upgrading your module to be Drupal 10 compatible](#)
3. [Import your module into your Developer Portal](#)
4. [Verifying that your module is Drupal 10 compatible](#)

## Installing the Upgrade Status module

1. Log in to your Developer Portal as the administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Install the Upgrade Status module. For more information about how to install the Upgrade Status, see [Installing custom modules](#).
4. When you finish your Upgrade Status installment, click Admin > Reports > Upgrade-status.

## Drupal 10 upgrade status ☆

Home » Administration » Reports

⚠ There is a security update available for your version of Drupal. To ensure the security of your server, you should update immediately! See the [available updates](#) page for more information and to install your missing updates.  
There are security updates available for one or more of your modules or themes. To ensure the security of your server, you should update immediately! See the [available updates](#) page for more information and to install your missing updates.

Run the report to find out if there are detectable compatibility errors with the modules and themes installed on your site.

GATHER DATA	FIX INCOMPATIBILITIES	RELAX
<ul style="list-style-type: none"> <li>Install <a href="#">Composer Deploy</a> or <a href="#">Git Deploy</a> as appropriate for accurate update recommendations</li> <li>Check available updates (Last checked 1 hour 1 minute ago)</li> <li>Scan: 7 projects</li> <li>Select any of the projects to rescan as needed below</li> </ul>	<ul style="list-style-type: none"> <li>Environment is incompatible</li> <li>Remove: 42 projects</li> <li>Update: 41 projects</li> <li>Collaborate with maintainers: 33 projects</li> <li>Fix with rector: 1 project</li> <li>Fix manually: 3 projects</li> </ul>	<ul style="list-style-type: none"> <li>Compatible with next major Drupal core version: 50 projects</li> </ul>

### ▼ DRUPAL CORE AND HOSTING ENVIRONMENT

Upgrades to Drupal 10 are supported from Drupal 9.4.x and Drupal 9.5.x. It is suggested to update to the latest Drupal 9 version available. Several hosting platform requirements have been raised for Drupal 10.

REQUIREMENT	STATUS
Drupal core should be at least 9.4.x	⚠ Version 9.4.8 allows to upgrade but 9.5.4 is available.
PHP version should be at least 8.1.0. Before updating to PHP 8, use <code>composer why-not php 8.1</code> to check if any projects need updating for compatibility. Also check custom projects manually.	⚠ Version 8.0.27
Database JSON support required	✓ Supported.
Invalid permissions will trigger runtime exceptions in Drupal 10. Permissions should be defined in a <code>permissions.yml</code> file or a permission callback.	<ul style="list-style-type: none"> <li>Permissions of user role: "Anonymous user": <ul style="list-style-type: none"> <li>search content</li> <li>use advanced search</li> <li>use search autocomplete</li> </ul> </li> <li>Permissions of user role: "Authenticated user": <ul style="list-style-type: none"> <li>search content</li> <li>use advanced search</li> <li>use search autocomplete</li> </ul> </li> <li>Permissions of user role: "Content Author": <ul style="list-style-type: none"> <li>access printer-friendly version</li> <li>manage content access control</li> </ul> </li> </ul>

5. The DRUPAL CORE AND HOSTING ENVIRONMENT section can be ignored. This area is managed by IBM API Connect and gets resolved when you upgrade.

Note: Your custom modules and themes must be PHP 8.1 compatible before you upgrade to Drupal 10, and you are responsible for doing the upgrade.

6. The list of projects further on the page are the ones that your Upgrade Status believes are not compatible with Drupal 10. These projects are categorized into sections, for example: REMOVE, UPDATE, COLLABORATE WITH MAINTAINERS.

When you upgrade IBM API Connect, the Drupal 10 compatibility is automatically given to the modules and themes that you have not created or installed manually.

### ▼ REMOVE

The likely best action is to remove projects that are uninstalled. Why invest in updating them to be compatible if you are not using them?

<input type="checkbox"/>	PROJECT	TYPE	STATUS	LOCAL VERSION	LOCAL 10-READY	LOCAL SCAN RESULT	DRUPAL.ORG VERSION	DRUPAL.ORG 10-READY	DRUPAL.ORG ISSUES	PLAN
<input type="checkbox"/>	AddToAny (addtoany)	Contributed module	Uninstalled	8.x-1.16	⚠ Incompatible	N/A	Not applicable	⚠ Unchecked	<a href="#">Issues</a>	N/A
<input type="checkbox"/>	APIC Test (apicest)	Contributed module	Uninstalled	8.x-5.2.3	⚠ Incompatible	N/A	Not applicable	⚠ Unchecked	<a href="#">Issues</a>	N/A
<input type="checkbox"/>	Autocomplete Deluxe (autocomplete_deluxe)	Contributed module	Uninstalled	2.0.2	⚠ Incompatible	N/A	Not applicable	⚠ Unchecked	<a href="#">Issues</a>	N/A
<input type="checkbox"/>	Back To Top (back_to_top)	Contributed module	Uninstalled	2.0.0	⚠ Incompatible	N/A	Not applicable	⚠ Unchecked	<a href="#">Issues</a>	N/A
<input type="checkbox"/>	Bootstrap Site Alert (bootstrap_site_alert)	Contributed module	Uninstalled	8.x-1.9	⚠ Incompatible	N/A	Not applicable	⚠ Unchecked	<a href="#">Issues</a>	N/A
<input type="checkbox"/>	Contact storage (contact_storage)	Contributed module	Uninstalled	8.x-1.2	⚠ Incompatible	N/A	Not applicable	⚠ Unchecked	<a href="#">Issues</a>	N/A
<input type="checkbox"/>	Serialization (CSV) (csv_serialization)	Contributed module	Uninstalled	8.x-2.1	⚠ Incompatible	N/A	Not applicable	⚠ Unchecked	<a href="#">Issues</a>	N/A
<input type="checkbox"/>	Devel (devel)	Contributed module	Uninstalled	4.2.1	⚠ Incompatible	N/A	Not applicable	⚠ Unchecked	<a href="#">Issues</a>	N/A
<input type="checkbox"/>	Disable Language (disable_language)	Contributed module	Uninstalled	8.x-1.0-beta11	⚠ Incompatible	N/A	Not applicable	⚠ Unchecked	<a href="#">Issues</a>	N/A
<input type="checkbox"/>	Drupal Proxy (droxy)	Contributed module	Uninstalled	2.0.1	⚠ Incompatible	N/A	Not applicable	⚠ Unchecked	<a href="#">Issues</a>	N/A

7. If you find your custom modules and themes there, you can scan them to reveal what changes are needed to make them Drupal 10 compatible. For instance, the following image demonstrates the process of scanning an installed custom module, named `custom_icons`.

<input type="checkbox"/>	Warmer (warmer)	Contributed module	Installed	2.0.8	⚠ Incompatible	N/A	Up to date	⚠ Incompatible	<a href="#">Issues</a>	N/A
<input type="checkbox"/>	IBM APIC Portal - custom icons (custom_icons)	Contributed module	Installed	1.0.0	⚠ Incompatible	5 problems	Not applicable	⚠ Unchecked	<a href="#">Issues</a>	N/A
<input type="checkbox"/>	connect_theme (connect_theme)	Contributed theme	Installed	8.x-5.2.5	⚠ Incompatible	N/A	Not applicable	⚠ Unchecked	<a href="#">Issues</a>	N/A

8. Select your custom modules and themes and then click Scan selected. When completed, if your custom module or theme requires changes, you see a link on the row of them.

Note: For Upgrade Status to identify all of the Drupal 10 compatibility issues in your custom modules and themes, you must have them enabled before scanning it.

**Fix now with automation**

Avoid some manual work by using [drupal-rector to fix issues automatically](#) or [Upgrade Rector to generate patches](#).

FILE NAME	LINE	ERROR
sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons/custom_icons.module	12	Call to deprecated function <a href="#">drupal_get_path()</a> . Deprecated in drupal:9.3.0 and is removed from drupal:10.0.0. Use Drupal\Core\Extension\ExtensionPathResolver::getPath() instead.
sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons/custom_icons.module	24	Call to deprecated function <a href="#">drupal_get_path()</a> . Deprecated in drupal:9.3.0 and is removed from drupal:10.0.0. Use Drupal\Core\Extension\ExtensionPathResolver::getPath() instead.
sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons/custom_icons.module	34	Call to deprecated function <a href="#">drupal_get_path()</a> . Deprecated in drupal:9.3.0 and is removed from drupal:10.0.0. Use Drupal\Core\Extension\ExtensionPathResolver::getPath() instead.
sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons/custom_icons.module	46	Call to deprecated function <a href="#">drupal_get_path()</a> . Deprecated in drupal:9.3.0 and is removed from drupal:10.0.0. Use Drupal\Core\Extension\ExtensionPathResolver::getPath() instead.

**Check manually**

Errors without Drupal source version numbers including parse errors and use of APIs from dependencies.

- Click export as HTML to see a clear view.

**Upgrade Status report****Contributed projects****IBM APIC Portal - custom icons 1.0.0**

Scanned on Thu, 03/02/2023 - 15:03.

4 errors found. 1 warning found.

**Fix now with automation**

Avoid some manual work by using [drupal-rector to fix issues automatically](#) or [Upgrade Rector to generate patches](#).

File name	Line	Error
sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons/custom_icons.module	12	Call to deprecated function <a href="#">drupal_get_path()</a> . Deprecated in drupal:9.3.0 and is removed from drupal:10.0.0. Use Drupal\Core\Extension\ExtensionPathResolver::getPath() instead.
sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons/custom_icons.module	24	Call to deprecated function <a href="#">drupal_get_path()</a> . Deprecated in drupal:9.3.0 and is removed from drupal:10.0.0. Use Drupal\Core\Extension\ExtensionPathResolver::getPath() instead.
sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons/custom_icons.module	34	Call to deprecated function <a href="#">drupal_get_path()</a> . Deprecated in drupal:9.3.0 and is removed from drupal:10.0.0. Use Drupal\Core\Extension\ExtensionPathResolver::getPath() instead.
sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons/custom_icons.module	46	Call to deprecated function <a href="#">drupal_get_path()</a> . Deprecated in drupal:9.3.0 and is removed from drupal:10.0.0. Use Drupal\Core\Extension\ExtensionPathResolver::getPath() instead.

**Check manually**

Errors without Drupal source version numbers including parse errors and use of APIs from dependencies.

File name	Line	Error
sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons/custom_icons.info.yml	0	Value of core_version_requirement: '^8    ^9' is not compatible with the next major version of Drupal core. See <a href="https://drupal.org/node/3070887">https://drupal.org/node/3070887</a> .

## Upgrading your module to be Drupal 10 compatible

This module is an example of how you can read the report and fix your code to be compatible with Drupal 10.

- The report shows that [drupal\\_get\\_path\(\)](#) is deprecated, and the code must be changed to use the newer function [getPath\(\)](#). The following examples demonstrate the change to the newer extension type-specific version.

**Before**

```
$name = preg_replace("/[^\A-Za-z0-9]/", '', $returnValue);
$returnValue = Url::fromUri('internal:/' . drupal_get_path('module', 'custom_icons') . '/images/' .
_custom_image_getRandomImageName($name))
->toString();
```

**After**

```
$name = preg_replace("/[^\A-Za-z0-9]/", '', $returnValue);
$iconPath = \Drupal::service('extension.list.module')->getPath('custom_icons');
$returnValue = Url::fromUri('internal:/' . $iconPath . '/images/' . _custom_image_getRandomImageName($name))
->toString();
```

- Add the new version into the `info.yml` file.
- `Drupal/Core/Extension/ExtensionList` is still present in Drupal 8 and Drupal 9, so you can keep them, and add on support for Drupal 10.
- Modify the version number when you make a new change to the module as seen in the following example.

**Before**

```
name: 'IBM APIC Portal - custom icons'
type: module
description: 'IBM API Connect Developer Portal tutorial - Example of custom product icons'
package: 'Custom'
core_version_requirement: ^8 || ^9
```

```
version: 1.0.1
project: 'custom_icons'
dependencies:
 - ibm_apim
```

After

```
name: 'IBM APIC Portal - custom_icons'
type: module
description: 'IBM API Connect Developer Portal tutorial - Example of custom product icons'
package: 'Custom'
core_version_requirement: ^8 || ^9 || ^10
version: 1.0.2
project: 'custom_icons'
dependencies:
 - ibm_apim
```

## Import your module into your Developer Portal

Import the module by using the Developer Portal [Toolkit CLI](#). For more information about how to use the Developer Portal CLI, see [Getting started with the Portal CLI commands](#).

For example,

```
> apic login -s $SERVER -r provider/default-idp-2 -u steve
Warning: Using default toolkit credentials.
Enter your API Connect credentials
Password?
Logged into api.fyre-ci-134466-master.fyre.ibm.com successfully

> apic -s $SERVER -m portaladmin custom-module:create-import -o ibm -c api-connect-catalog-2 /tmp/custom_icons.zip
Loading File (Large files may take a while)...

201 CREATED - Task ID: wp5fhmm9174pmkw1

Response Code - 202:

Message(s) -
The status of this task is: QUEUED

Response Code - 202:

Message(s) -
The status of this task is: RUNNING

Response Code - 200:

Message(s) -
2023-02-27 14:39:38: CLI task (custom_module:import) starting.
2023-02-27 14:39:40: Deleting existing custom module /var/aegir/platforms/devportal-9.x-10.0.5.2-20230105-1720/sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons
2023-02-27 14:39:40: Importing custom module custom_icons
2023-02-27 14:40:25: CLI task (custom_module:import) completed successfully.
```

Alternatively, import your updated module by installing it again using the Developer Portal UI. For more information about how to install custom modules, see [Installing custom modules](#).

## Verifying that your module is Drupal 10 compatible

The updated module is now imported into the Developer Portal. Return to the Upgrade Status report page to rescan the module. When it becomes compatible with Drupal 10, it appears at the end of the page.

<input type="checkbox"/>	Workbench (workbench)	Contributed module	Installed	8.x-1.4	✓ Compatible	N/A	Up to date	✓ Compatible	<a href="#">Issues</a>	N/A
<input type="checkbox"/>	IBM APIC Portal - custom icons (custom_icons)	Contributed module	Installed	1.0.0	✓ Compatible	No problems found	Not applicable	ⓘ Unchecked	<a href="#">Issues</a>	N/A
<input type="checkbox"/>	Snowflakes (snowflakes)	Contributed module	Uninstalled	2.0.1	✓ Compatible	3 problems	Not applicable	ⓘ Unchecked	<a href="#">Issues</a>	N/A

## Results

You used the Upgrade Status module to identify and fix the Drupal 10 compatibility problems with your custom modules and themes. However, this process might not cover everything that you need to be ready for the upgrade to Drupal 10. Drupal 10 requires PHP 8.1, so you might need to change your custom modules and themes to support it. Remember to keep your custom modules and themes up to date, and to remove deprecated code within your own written custom modules and themes.

## Upgrade your custom modules to be Drupal 10 and PHP 8.1 compliant

Guidance on how to upgrade your custom modules to be Drupal 10 and PHP 8.1 compliant.

The following sections provide examples of the most common updates that you might need to make. However, you must check the deprecation lists for Drupal and PHP to ensure that you cover all of the scenarios in your Developer Portal sites.

## Notable changes the update brings

The following items are some of the key changes in Drupal 10.

- **CKEditor4** is removed and **CKEditor5** is now the default text editor.
- **Claro** is now the default theme for the Admin UI, which gives it a modern and refreshing look.
- Ensure that none of your custom modules depend on them as they will not be there after the upgrade.

The following modules are uninstalled as they do not support Drupal 10:

- ckeditor\_media\_embed
  - console
  - flood\_unblock
  - message
  - message\_notify
  - message\_subscribe
  - video\_embed\_field
  - video\_embed\_html5
  - content\_browser
  - entity\_browser
  - entity\_browser\_enhanced
  - entity\_embed
  - file\_browser
  - editor\_file
  - color
  - Unlimited\_number
  - embed
  - ckeditor
  - quickedit
- The Drupal 10 upgrade comes with a major Symfony upgrade version. Drupal 9 used Symfony 4.4, and Drupal 10 now uses Symfony 6, so if you are using any Symfony ensure that they have not been deprecated. Symfony provides a guide at the following link [https://symfony.com/doc/current/setup/upgrade\\_major.html](https://symfony.com/doc/current/setup/upgrade_major.html).
  - The default Drush version is now Drush 11. If you are writing Drush commands in your custom modules, you must ensure they are Drush 11 compatible. The following links provide examples on how you can achieve compatibility, <https://www.specbee.com/blogs/writing-your-own-custom-drush-9-and-10-commands> and <https://www.drush.org/11.x/commands/>.

## PHP 8.1 deprecations

---

The following items are some of the key PHP 8.1 deprecations.

Passing null to non-nullable internal function parameters is deprecated

There are multiple scenarios that require a fix with the possibility to pass a `null` to some PHP functions.

The following example is the most common PHP deprecation.

**Before:**

```
$redacted_url = $url;
if (strlen($redacted_url) > 0 && strpos($redacted_url, '/') !== 0) {
```

**After:**

```
$redacted_url = $url ?? '';
if (strlen($redacted_url) > 0 && strpos($redacted_url, '/') !== 0) {
```

Optional parameters specified before the required parameters

Some functions in code have optional parameters before the required ones. However, in PHP 8.1, the required parameters come after the optional ones.

Improved type safety

PHP 8.1 improves the type of safety, which can result in numerous warnings that might require changing functions and variable declarations.

The following example addresses the improved safety and the optional parameters before required parameters deprecations.

**Before:**

```
protected $id;

public function oldFunction($url = '', $data) {
 return $id;
}
```

**After:**

```
/**
 * Drupal entity ID
```

```

*
* @var string
*/
protected $id;

public function newFunction(someObject $data, string $url = ''): string {
 return $id;
}

```

## Drupal 10 deprecations

The following items are some of the key PHP 8.1 deprecations.

Entity queries now require an `accessCheck`

If you have custom modules with entities, you must do an `accessCheck`. For more information about access checking, see <https://www.drupal.org/node/3201242>.

For example,

**Before:**

```

$query = \Drupal::entityQuery('node');
$query->condition('type', 'api');
$query->condition('status', 1);
$nids = $query->execute();

```

**After:**

```

$query = \Drupal::entityQuery('node');
$query->condition('type', 'api');
$query->condition('status', 1);
$nids = $query->accessCheck()->execute();

```

The `once` function is removed from the jQuery library and added to its own core library

If you have any modules that use the `once` function in your JavaScript files, you need an update to pull in the new library. For more information about how to remove jQuery dependency, see <https://www.drupal.org/node/3158256>.

For more examples on how you might upgrade your code, see <https://www.drupal.org/project/drupal/issues/3183149>.

For example,

**Before:**

```

// ibm_apim.libraries.yml
validate_password:
 version: 1.x

// validate_password.js
(function($, Drupal, drupalSettings) {
 var $passwordInput = $(context).find('input.js-password-field').once('ibmApimValidatePassword');
})(jQuery, Drupal, drupalSettings);

```

**After:**

```

// ibm_apim.libraries.yml
validate_password:
 version: 1.x
 dependencies:
 - core/once

// validate_password.js
(function($, Drupal, drupalSettings, once) {
 var $passwordInput = $(once('ibmApimValidatePassword', context)).find('input.js-password-field');
})(jQuery, Drupal, drupalSettings, once);

```

The `app.root` and `site.path` services that are converted into container parameters

If you use the `app.root` and `site.path` services in your custom modules, you must ensure they are converted into container parameters.

For example,

**Before:**

```

\Drupal::service('config.factory')
->getEditable('locale.settings')
->set('translation.use_source', 'local')
->set('translation.path', \Drupal::service('site.path') . '/translations')
->save();

```

**After:**

```

\Drupal::service('config.factory')
->getEditable('locale.settings')
->set('translation.use_source', 'local')
->set('translation.path', \Drupal::getContainer()->getParameter('site.path') . '/translations')
->save();

```

The parameters of some `Forms` are updated, so be aware if you are extending them.

If you are extending your `Forms`, note the parameters of some of them have changed. For more information about the updated parameters, see

<https://www.drupal.org/node/3251987>.

## Next steps

The sections above cover the key upgrades you might need when upgrading your custom modules to be Drupal 10 and PHP 8.1 compliant. However, note that it is your responsibility to check the deprecation lists for Drupal and PHP to ensure that you cover all of the scenarios in your Developer Portal sites.

For more information about making your custom modules Drupal 10 and PHP 8.1 compliant, see the following links:

- [Deprecation checking and correction tools.](#)
- [Deprecated modules and themes.](#)
- [List of APIs deprecated and now removed from Drupal 10.](#)
- [Whats new in PHP 8.1.](#)
- [Symfony Upgrade Guide.](#)
- [Drush 11 commands.](#)

---

## Developer Portal tutorials

Tutorials for using the Developer Portal.

The following guided examples help you through some of the most common user scenarios in the Developer Portal, as well as some of the more complex areas.

Developer Portal tutorials by category
For API consumers <a href="#">Create an Application</a>
User authentication <a href="#">Adding a field to the sign up form</a> <a href="#">Adding validation to a field on the sign up form</a> <a href="#">Adding a field group to group fields in a content type</a>
Getting started configuring the Developer Portal <a href="#">Creating the Portal</a> <a href="#">Creating a private Portal site</a> <a href="#">Customizing themes</a>
Advanced customization <a href="#">Changing the front page</a> <a href="#">Grouping products by category</a> <a href="#">Adding a custom block to the front page</a> <a href="#">Adding a custom block to a page other than the front page</a> <a href="#">Using avatars</a> <a href="#">Using image optimization</a> <a href="#">Adding a Back to top button</a> <a href="#">Creating a drop-down menu link</a> <a href="#">Displaying blog posts on the front page</a> <a href="#">Displaying tweets on the front page</a> <a href="#">Changing the profile page to display firstname lastname, instead of username</a> <a href="#">Using a custom weighting sort order on the product list page</a> <a href="#">Configuring the RobotsTxt file</a> <a href="#">Configuring the SecurityTxt file</a> <a href="#">Synchronizing application credentials with an external server</a>

---

## Tutorial: Creating the Developer Portal

### About this task


---

In this tutorial, you are going to complete the following lessons:

### Enabling the Developer Portal through the API Manager

---

By proceeding with the following steps, you create an administrator account:

1. Log in to the API Manager
2. Click  Manage, then click Add, and select Create catalog.
3. Select the owner from the **Select user** list, then enter a **Title** for your catalog, for example `Production`.



# Create Catalog

### Create Catalog

Enter the catalog summary details; you can fully configure the catalog after you create it

---

Select user

Apic Tutorial (tutor), myuser@myco.com

---

Title

Production


---

Name

production


---


Cancel Create

- Click Create, then click  Manage.

## Manage

A catalog hosts a collection of API products that are visible in the associated developer portal when published

 Production

 Sandbox

Sandbox Catalog

- Click your new catalog, for example **Production**.
- Click Catalog settings.
- Click Gateway Services. If the message **You currently don't have any gateway services** is displayed, click Edit, select gateway\_service, then click Save.

## Enable Gateway Services

<input checked="" type="checkbox"/>	Title	Type
<input checked="" type="checkbox"/>	gateway_service	DataPower API Gateway

---

Cancel Save

- Click Portal.
- Click Create.

# Settings

Overview

Gateway Services

Lifecycle Approvals

Roles

Onboarding

API User Registries

OAuth Providers

API Endpoints

TLS Client Profiles

**Portal**

Catalog Properties

Portal Create

! No portal added to this catalog X

10. Select a portal service to use. Click Create.

## Create Portal

Portal Site

Create a portal site for the catalog

Select the portal service to use for this catalog

portal\_service

URL (optional)

Cancel Create

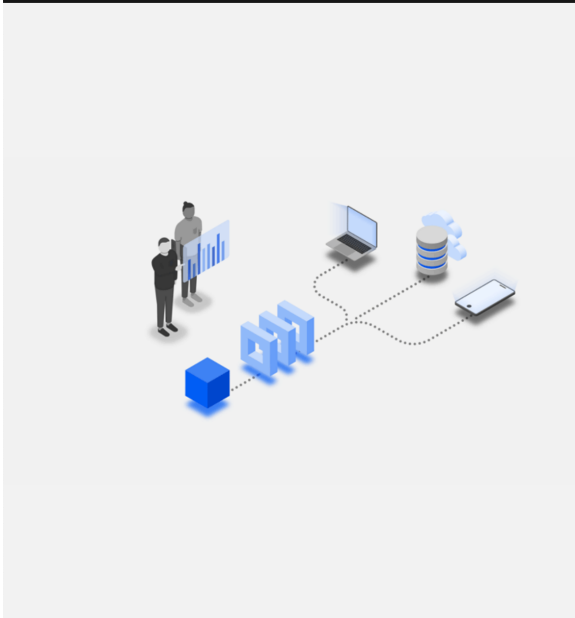
After a few minutes, you receive an email with a link to your Developer Portal site for that catalog. The link is a single use only link for the administrator account.

Administrator (admin),

Your Developer Portal site has been created. You can log in as the 'admin' user by using the following one-time log in link. After you have logged in with the following link, you must change the password for the 'admin' user account immediately.

[https://portal.apidemo.demo.ibm.com/tutorial/production/user/reset/1/1591727237/h26vxzf2e51UqgK7vf10LNbHiVEC5VQ\\_AZAoKCYH72C/new/login](https://portal.apidemo.demo.ibm.com/tutorial/production/user/reset/1/1591727237/h26vxzf2e51UqgK7vf10LNbHiVEC5VQ_AZAoKCYH72C/new/login)

11. Click the email link. Click Sign in.



API Developer Portal

## Reset password

If you have forgotten the password for your account or 'admin', you can reset it here. This is a one-time login for *admin*.

Click on this button to log in to the site and change your password.

This login can be used only once.

Sign in

[Back to Sign in](#)

12. Change the password. Click Submit.

✔ You have just used your one-time login link. It is no longer necessary to use this link to log in. Please change your password. ✕

## Change your password

Change your password.

Password \*

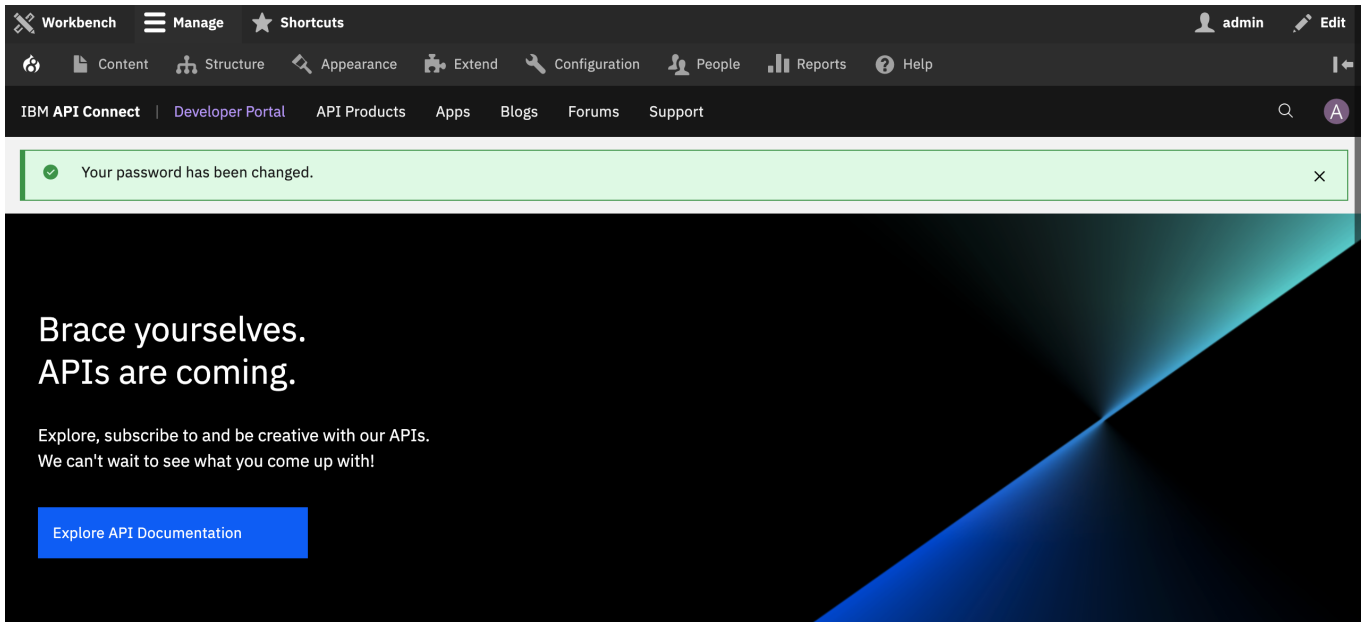
Password strength:

Confirm password \*

- Minimum password character types: 3
- Password character length of at least 8 characters
- Password must not contain the user's username.
- Password Strength minimum score of 2
- Maximum consecutive identical characters: 3

Submit

The portal site for the Production catalog is now active. The admin account cannot create applications or subscriptions. You must create a new account to complete those tasks.



Let's get started!

Sign up

[Create a new account](#) and get started with our APIs. It's free to

Explore our APIs

Take a look at our [API products](#) and quickly find APIs to construct

Create

Subscribe to a plan and create your application to make use of

## What you did in this tutorial

In this tutorial, you completed the following activities:

- Enabled a Developer Portal site and created an administrator account.

## What to do next

- [Create a Developer Account and Application](#)

## Related information

- [Importing an API](#)

## Tutorial: Creating a private Developer Portal site

You can create a private version of the Developer Portal, so that authenticated users only can access content on the site. You might want to do this if you want to restrict public access to your content unless they are logged in.

## Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial [Creating the Portal](#) explains how to enable the portal if you have not already done so.

## About this tutorial

In this tutorial, you are going to restrict access for all the content to authenticated users only. This gives access to all users from **Content Author** to **Superuser**, as they are all required to authenticate to access their roles. You will then add the login form to the home page, so that public users will be able to view the public home page that contains the login form only.

This tutorial takes you through the following steps:

1. [Export your configuration settings.](#)
2. [Restrict all permissions to authenticated users.](#)
3. [Restrict blocks to authenticated users.](#)
4. [Add login form to the home page.](#)

## Export your configuration settings

In case you make an error, or want to revert all of the changes made during this tutorial, you can back up your configuration settings to your machine before you make any changes.

The tutorial [Exporting and importing custom themes and site configuration](#) provides further details on how to export your configuration settings using the Developer Portal CLI.

## Restrict all permissions to authenticated users

---

1. If the administrator dashboard isn't displayed, click **Manage** to display it.
2. Navigate to **People > Permissions**.
3. Clear all check boxes that are associated with **ANONYMOUS USER**.
4. Click **Save permissions** at the end of the page.

## Restrict blocks to authenticated users

---

1. If the administrator dashboard isn't displayed, click **Manage** to display it.
2. Navigate to **Structure > Block layout**.
3. Find the row that is labeled **Main menu** and click **Configure**.
4. Find the setting **Roles** under the category **Visibility**.
5. Check the box **Authenticated user**.
6. Click **Save block**.
7. Repeat from step 3 for the **Footer menu**.
8. Click **Save blocks**.

## Add login form to the home page

---

1. If the administrator dashboard isn't displayed, click **Manage** to display it.
2. Navigate to **Structure > Pages**.
3. Click **Edit** on the **Welcome** page.
4. Navigate to **Panels > Content**.
5. Click + **Add new block**.
6. Find and click **User login**, under the **FORMS** section.
7. Clear the **Display title** check box.
8. Click **Add block**.
9. Rearrange the new login block so that it is underneath the **Welcome Banner**.
10. Click **Update and save**.

## What you did in this tutorial

---

On your Developer Portal, you restricted access for all the content to authenticated users only. Now when you access your home page as an unauthenticated user, you see that the login form appears underneath the welcome banner. Navigating to other URLs on the site presents an **unauthorized** page to appear.

When you log in, the login form disappears from the home page and you are able to access all of the pages that are visible to your role.

---

## Tutorial: Creating Accounts and Applications on the Developer Portal

This tutorial shows you how to create a developer account and register an application on the Developer Portal.

### Before you begin

---

You must have a Developer Portal installed, and created and published at least one API product.

If you do not have a Developer Portal available, complete the tutorial [Creating a Developer Portal](#).

If you don't have an API product available, you can download one of the following APIs, and then import and activate it:

- If your Sandbox catalog uses a DataPower® Gateway (v5 compatible) (as one of the gateways, or as the only gateway), download the [findbranch.txt](#) file to your local filesystem. Rename this file `findbranch.yaml`.
- If your Sandbox catalog uses a DataPower API Gateway, download the [findbranch\\_v6.txt](#) file to your local filesystem. Rename this file `findbranch.yaml`.

For more information about importing an API, see [Adding a REST API by importing an OpenAPI definition file](#).

### About this task

---

In this tutorial you are going to complete the following lessons:

- [Creating a developer account in the Developer Portal](#)
- [Creating a developer application in the Developer Portal](#)
- [Testing the Branches API in the Developer Portal](#)

Note: The administrator account cannot create an App, or register to an App. To create or register to an App, you must have a developer account. A developer account that has been assigned administrator privileges can create and register Apps.

## Creating a developer account in the Developer Portal

---

Take the following steps to create a developer account:

1. Open a Developer Portal at the previously specified URL, then click Create an account and complete the fields. If the Create an account link is not visible, log out as Admin from the Developer Portal first.  
Note: Your email is used as your user name for the Developer Portal. The email address must be different from the address used for creating the administrator account.
2. Go to the home page of the Developer Portal
3. Click Create an account.
4. Complete the sign up form. The email address you provide must be functional. Click Sign up. A confirmation message is sent to the email address provided in the form.

5. Click the account activation link in the confirmation email to activate your account.
6. To use the Developer Portal, click Login and sign in with the user credentials you specified.

## Creating a Developer Portal Application

To create and register a new App:

1. In the Developer Portal, click Create an App.

Note: You cannot register new applications if you are logged in as the administrator.

2. Enter the following values for the application that is being registered.

Table 1. Values for registering the application

Field Name	Value
Title	AppOne
Application OAuth Redirect URL(s)	http://example.com

### Create a new application

Title \*

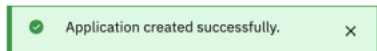
Description

Application OAuth Redirect URL(s)

[Add another item](#)

Cancel Save

- Click Save.
- Copy the key and secret values for the new app. Click OK.



### New application credentials

The API Key and Secret have been generated for your application.

Key

88b0233e5322a2c898be2793a9e9cbdc

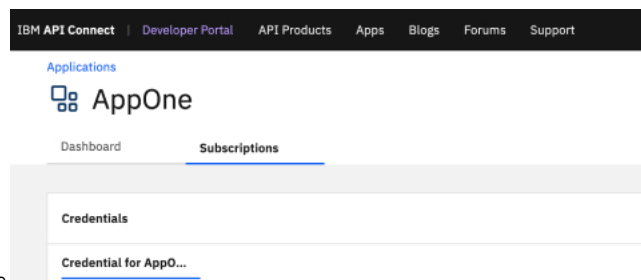
Secret

9a64e91f023a441d9a2d6ec6ab491fa4

The Secret will only be displayed here one time. Please copy your API Secret and keep it for your records.



- Click Continue.



You see a usage statistics page.

## Subscribing to an API in the Developer Portal

- Click API Products.
- Click FindBranch.
- Click Subscribe.
- Under the Application heading, click Select App for your new application.
- Click Next.
- Click Done.


## Testing An API in the Developer Portal

- Click API Products in the Developer Portal dashboard.
- Click a Product, such as FindBranch auto product , then select the FindBranch API from the list.
- Click GET /details.

Filter

- Overview
- GET /details
- Definitions

### Overview

Download Open API Document 

Type REST

4. Click Try it.

## GET : /details

Details Try it

**GET** Production, Development: [https://\[redacted\]/tutorial/sandbox/findbran](https://[redacted]/tutorial/sandbox/findbran)

Security

**clientID**  
X-IBM-Client-Id apiKey located in header

5. Click Send.

Details Try it

**GET** Production, Development: [https://\[redacted\]/tutorial/sandbox/findbran](https://[redacted]/tutorial/sandbox/findbran)

Security

**Identification**

**Client ID**  
AppOne

Parameters

Header

Accept  
application/json

Reset Send

Note: If no response is received, navigate to the URL that is displayed at the beginning of the Try this operation section, in a new browser tab. Accept the security certificate, and then call the operation again.

6. A returned response of 200 OK and the message body are displayed, indicating that the REST API operation call was successful.



[Reset](#) [Send](#)

---

Request

```
GET https://[REDACTED]/tutorial/sandbox/findbranch/details
Headers:
Content-Type: application/json
Accept: application/json
X-IBM-Client-Id: bdd47d45d7775a24e2326dc46dbb9dfe
```

Response

```
Code: 200 OK
Headers:
content-type: application/json; charset=utf-8
x-global-transaction-id: 196c55655b0307b1000b1da1
[
 {
 "id": "02e91060-4d49-11e8-a6f5-b75dbc3b13de"
 },
 {
 "id": "0b3a8cf0-7e78-11e5-8059-a1020f32cce5",
```

---

## What you did in this tutorial

In this tutorial, you completed the following activities:

- Created a developer account in the Developer Portal.
- Created and registered a new App, and subscribed it to a Plan.
- Tested an API in the Developer Portal.

---

## Tutorial: Creating a custom theme for the Developer Portal

Customize the appearance of your Developer Portal home page, by generating a sub-theme, configuring the overrides.css file, and installing the newly customized theme.

---

### Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial [Creating the Portal](#) explains how to enable the portal if you have not already done so.

Important: You're not permitted to include any IBM® API Connect code within any custom themes that you create.

---

### About this tutorial

The way to create a custom theme is to create a custom sub-theme of the standard API Connect theme that the Developer Portal uses by default. The sub-theme inherits all of the resources of the parent theme, and you can then override specific resources to configure the required customizations.

This tutorial takes you through the following lessons:

1. [Creating a sub-theme](#) by using the theme generator.
2. [Customizing the overrides.css file](#).
3. [Installing and enabling your customized theme](#).

---

### Creating a sub-theme

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click **Manage** to display it.
3. Click **Appearance** > **Generate sub-theme**. The **Generate sub-theme** window is displayed.
4. Enter a Sub-theme name, and select **CSS** for the Sub-theme type. (If you prefer, you can select **SCSS**, but this extension to CSS is for advanced theme developers, and isn't covered by this tutorial.)

## Generate sub-theme ☆

List Generate Update Settings

Home » Administration

The first step in customizing the branding of your Developer Portal is to create a custom sub-theme.

The sub-theme inherits all of the resources of the parent theme, and you can then override specific resources in the overrides.css file to configure your customizations. For more information, see: [Knowledge Center](#)

Complete the form below and you will be presented with a custom sub-theme to download.

Sub-theme name \*

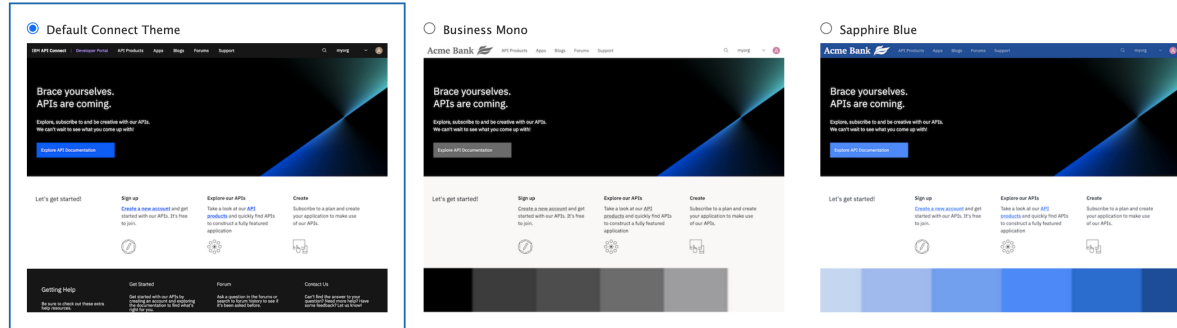
A custom theme name, for example: 'mycustom\_theme' or 'banka\_theme'. The name does not need to end in '\_theme' but it is a common convention.

Sub-theme type

- CSS  
 SCSS

Your sub-theme can be setup to use either CSS or SCSS. SCSS is an extension to CSS and is for more advanced theme developers.

Template



5. Select the Default template to base your **sub-theme** on. You can create a sub-theme based on a color template. However, for this tutorial you use the default connect\_theme template.
6. Click Generate.
7. Download the generated sub-theme to a location of your choice, and extract all the files from the .zip file.

## Customizing the overrides.css file

On your computer, navigate to the files that you extracted. Find the overrides.css file in here banka\_theme/banka\_theme/css/overrides.css. You can now customize the overrides.css file.

1. Open the overrides.css file in your chosen editor.
2. Customize your sub-theme by entering the following elements into the overrides.css file:

```
/* Body of the home page */
body.path-frontpage.contexthome {
 color: #152935;
 background-color: #d1f0f7;
}

/* Header of the home page */
.navbar {
 background-color: #a0a0a0;
}

/* Footer of the home page */
footer.footer {
 background-color: #dee0e2;
}

footer.footer ul.nav > li > a {
 color: #454A4C;
}
```

```
File Edit Format View Help
/* Body of the home page */
body.path-frontpage.contexthome {
 color: #152935;
 background-color: #d1f0f7;
}

/* Header of the home page */
.navbar {
 background-color: #a0a0a0;
}

/* Footer of the home page */
footer.footer {
 background-color: #dee0e2;
}

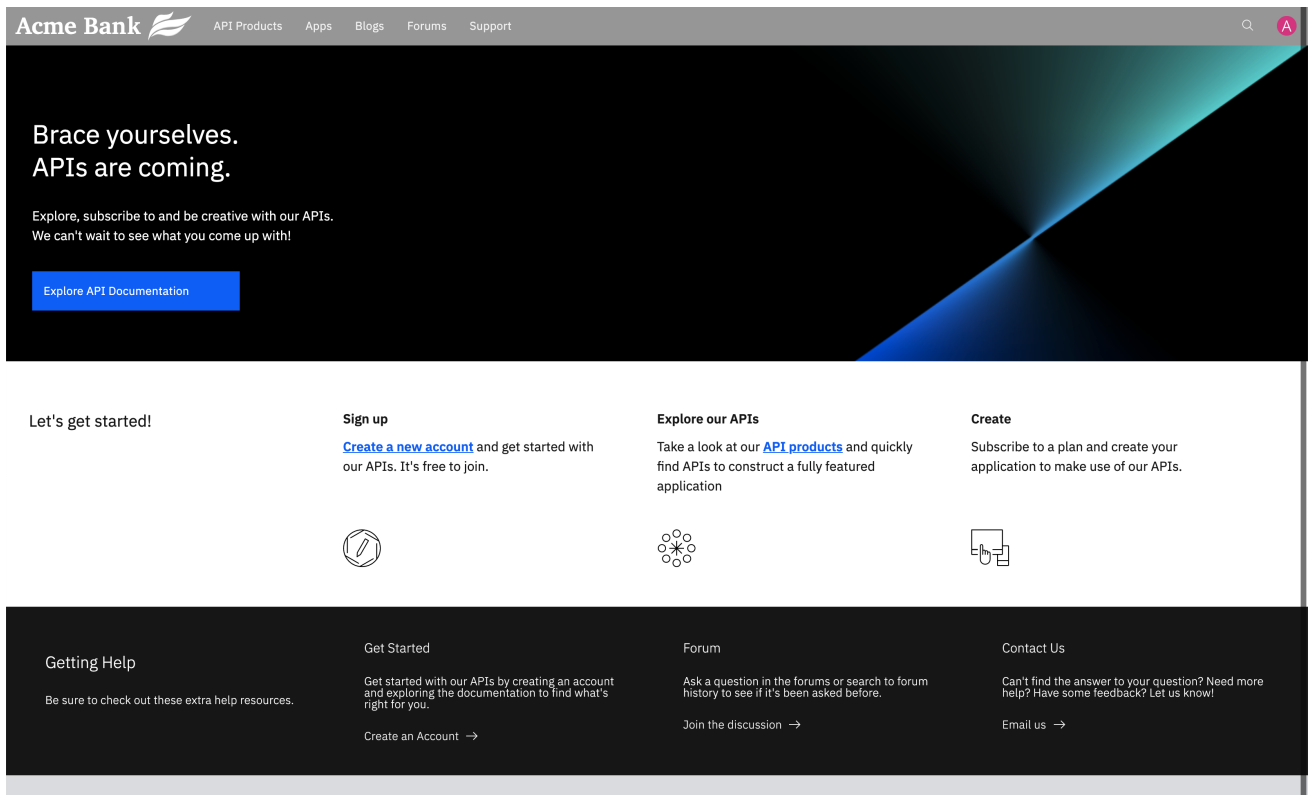
footer.footer ul.nav > li > a {
 color: #454A4C;
}

Ln 19, Col 2 100% Windows (CRLF) UTF-8
```

3. Save the overrides.css file.

## Installing and enabling your customized theme

1. After you have finished updating the overrides.css file, compress all the theme files back into the .zip sub-theme file that you downloaded originally. Note that you can also include your own custom logo and favicon in your theme files, and these are then included as part of your theme when it is installed. These files must use the naming convention logo.svg, and favicon.ico, respectively. You can also change your logo and favicon at a later time, see [Changing the site logo](#) and [Changing the shortcut icon](#).
2. In the Developer Portal, click Appearance > Install new theme. The **Install new theme** window is displayed.
3. In Upload a module or theme archive to install click Browse, and navigate to your newly updated compressed theme file.
4. Click Install to install the theme onto your site.
5. Click Enable newly added themes, and find your new theme in the list of Disabled themes. Click Enable and set as default to set your new custom sub-theme as the default theme for your site.
6. Return to the Developer Portal home page by clicking Back to site. You can now see your custom theme.



## What you did in this tutorial

In this tutorial, you completed the following tasks:

- Created a sub-theme of your Developer Portal site by using the theme generator.
- Configured some customized home page elements in the overrides.css file of your sub-theme.
- Uploaded and installed your customized theme in your Developer Portal site.
- Successfully customized the appearance of your Developer Portal home page.

## Tutorial: Using avatars

You can use an avatar generator to better represent the branding of your site, and add a sense of community for your users, than the default name initial that is provided in the Developer Portal.

### Before you begin

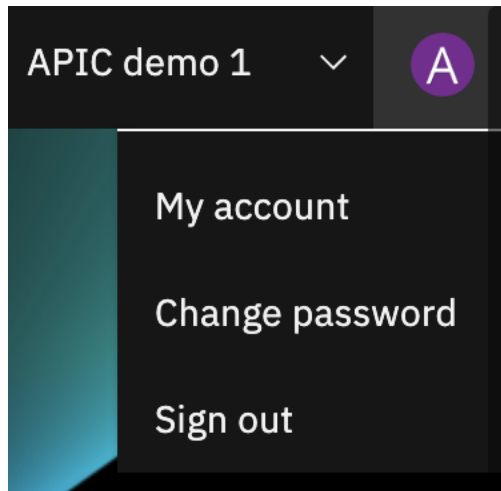
You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial [Creating the Portal](#) explains how to enable the portal if you have not already done so.

### About this tutorial

In this task, you add the **Robohash** avatar generator to your Developer Portal site, and assign it as the primary avatar generator for all users.

This tutorial takes you through the following steps:

1. [Install the Robohash avatar generator.](#)
2. [Set Robohash as the preferred avatar generator for your site.](#)
3. [Configure the permissions for the Robohash avatar generator.](#)



The avatar of the admin user displays like this before the block is added.

## Enable the Robohash avatar generator

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Extend.
4. In the text box, enter `avatar` in the **Block description**.
5. Check the **Robohash** box, click Enable.

+ Install new module

Enter a part of the module name or description

Below is a list of installed modules. Modules that are installed and enabled are shown with a selected check box. To enable a module,

### ▼ AVATARS

<input type="checkbox"/>	<b>Adorable</b>	▶ Adds an Adorable.io avatar generator for Avatar Kit.
<input checked="" type="checkbox"/>	<b>Avatar Kit</b>	▶ Provides a selection of generated avatars for users.
<input type="checkbox"/>	<b>Gravatar</b>	▶ Adds Gravatar.com avatar generators for Avatar Kit.
<input checked="" type="checkbox"/>	<b>Letter Avatar</b>	▶ Adds a letter avatar generator based on the name of a user.
<input checked="" type="checkbox"/>	<b>Robohash</b>	▶ Adds Robohash avatar generators for Avatar Kit.

Enable

When the module is enabled, you receive a **Module Robohash has been enabled** message.

## Set Robohash as the preferred avatar generator for your site

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Configuration > People > Avatar settings.
3. Click + **Add avatar generator**.
4. Enter `Robohash` as the label.
5. Check **Robohash** for the avatar generator.
6. Click Save

## Add avatar generator ☆

Label \*

Avatar generator \*

- APIC Letter Avatar**  
Letter generated by the ApicLetterAvatar library.
- Robohash**  
Robots and monsters from Robohash.org
- User preference**  
Avatar generator calculated based on user preference.
- User upload**  
Image uploaded by the user to the site.

Save

7. You are then taken to the **Edit avatar generator Robohash** page. Select **Type: Robot Heads** and **Background: Transparent**. Click Save.

Label \*

Robohash

Type

- Robots
- Robot Heads
- Monsters

Background

- Transparent
- Places
- Patterns

Save

Delete

8. You are now on the avatar generator page. Drag **Robohash** to be first in the list, and check **Enabled**, then click Save configuration.

You have now set **Robohash** as the preferred avatar generator for your site.

## Configure the permissions for the Robohash avatar generator

The default setting for the **Robohash** avatar generator is to generate avatars for users with **superuser** access only. In this example, the permissions are changed so that all **authenticated users** can access **Robohash**.

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to **People > Permissions**.
3. Find **Use Robohash** under the permission type **Avatar Kit**.
4. Check **AUTHENTICATED USER** for **Use Robohash** to enable the generator for all authenticated users.  
Notice that all the boxes for this user type are then automatically checked because all of those users have the same permissions as an authenticated user, or higher.

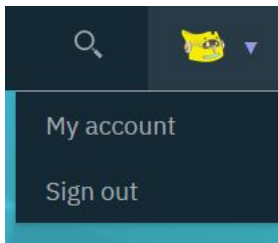
PERMISSION	ANONYMOUS USER	AUTHENTICATED USER	FORUM MODERATOR	CONTENT AUTHOR	ADMINISTRATOR	SUPERUSER
Change own logout threshold	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<b>Avatar Kit</b>						
Administer Avatar Kit <i>Warning: Give to trusted roles only; this permission has security implications. Manage settings for Avatar Kit.</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Use Letter Generator User can select Letter Generator avatar generator.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Use Robohash User can select Robohash avatar generator.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

5. Click Save permissions.

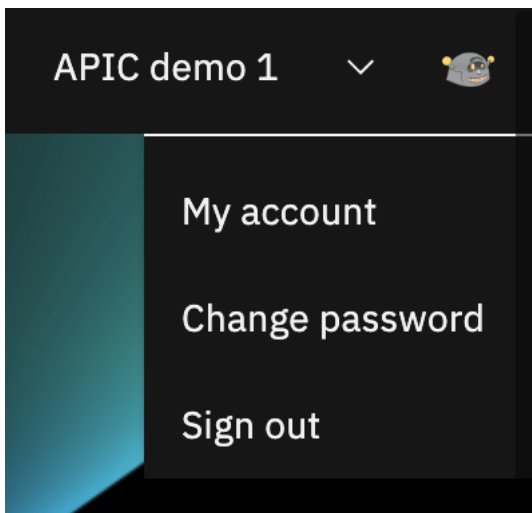
## What you did in this tutorial

The avatars of all users now have the **robohash** avatar.

You can check that the avatar of the admin user has changed by looking at your site page.



You can check that the avatars of other users have changed by finding their published content and looking for their avatar. Or, you can navigate to Manage > People, click a user, and see their avatar on their profile page.



## What to do next

You can disable the **robohash** avatar generator.

1. Navigating to Extend > Disable module.
2. Search for **avatar** in the filter text box.
3. Check **Robohash**, click Disable
4. Click Disable.

You can also disable the **robohash** avatar generator by clearing it within Configuration > People > Avatar settings, or, reducing its preference by dragging it to a different order in the list. Click Save configuration.

## Tutorial: Using image optimization

You can use image optimization on your Developer Portal site to enhance the use of images.

Optimized images do not overload your website and significantly increase its performance compared to using the original images. Your customer satisfaction and brand reputation improve with fast-loading and trimmed visual assets. Automatic optimization can save time and effort on editing and content moderation.

## Before you begin

---

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial [Creating the Portal](#) explains how to enable the portal if you have not already done so.

## About this tutorial

---

In this task, you create an image style that is called `User picture`, and give it a scale and crop effect. Then, use the `User picture` as the `Picture` field in the user account.

This tutorial takes you through the following steps:

1. [Create the image style `User picture`.](#)
2. [Edit the image style `User picture`.](#)
3. [Assign `User picture` to the `picture` option in `Account settings`.](#)

## Create the image style `User picture`

---

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click `Manage` to display it.
3. Navigate to `Configuration > Media > Image styles`.
4. Click `+` Add image style.
5. Enter `User picture` in the `Image style name` block. Click `Create new style`

### Add image style

[Home](#) » [Administration](#) » [Configuration](#) » [Media](#) » [Image styles](#)

Image style name \*

Machine name: `user_picture` [\[Edit\]](#)

[Create new style](#)

You created the image style `User picture`.

## Edit the image style `User picture`

---

When you edit this style, you choose the width and height settings. Also, the part of the picture to retain during the crop, for example, the center.

1. If the administrator dashboard isn't displayed, click `Manage` to display it.
2. Navigate to `Configuration > Media > Image styles`.
3. Click `Edit` on the `User picture` style name.



## Edit style *User picture* ☆

Edit

Translate image style

Home » Administration » Configuration » Media » Image styles

### Preview

original (view actual size)



600px

800px

### Image style name \*

User picture

Machine name: user\_picture [Edit]

### EFFECT



Select a new effect

Add

Save

Delete

4. Select **Scale and crop** from **Select a new effect**, then click Add.

5. Enter 300 in the **Width** and **Height** boxes. Leave the **Anchor** as the middle setting. Click Add effect.

Scale and crop will maintain the aspect-ratio of the original image, then crop the larger dimension.

### Width \*

300



pixels

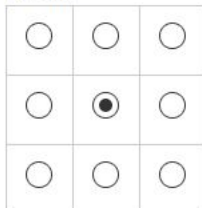
### Height \*

300



pixels

### Anchor



The part of the image that will be retained during the crop.

Add effect

Cancel

6. Click Save.

✓ Changes to the style have been saved.


You edited the image style **User picture**.

## Assign **User picture** to the **picture** option in **Account settings**

1. If the administrator dashboard isn't displayed, click Manage to display it.

2. Navigate to Configuration > People > Account settings > Manage display, then click Compact tab.

FIELD	LABEL	FORMAT	
Picture	- Hidden -	Image	Image style: Thumbnail (100x100) Linked to content

3. On the **Picture** row, click .
4. Select **User picture** from the **Image style** drop-down.
5. Click Update and Save.

You set the **User picture** image style for all accounts.

## What you did in this tutorial

You changed your account settings so that if the user displays a picture it is within the limits of the image style **User picture**.

To check the picture used by a user, navigate to Manage > People, click a user, and see their picture on their profile page. Right-click the image, then click **View image info**, you can see that the dimensions are scaled.

Type:	JPEG Image
Size:	29.95 KB (30,667 bytes)
Dimensions:	620px × 413px (scaled to 240px × 160px)
Associated Text:	Profile picture for user apicdemo1

## Tutorial: Adding a Back to top button

You might want to add a Back to top button, as a navigation enhancement to help your Developer Portal users get back to the top of the page that they are viewing, instead of scrolling.

### Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial.

### About this tutorial

In this tutorial, you add a Back to top button to your Developer Portal.

Before you begin, ensure that the **Back to top** module is installed and enabled:

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Extend.
4. In the text box, enter `back to top` in the **Block description**.
5. Check the **Back To Top** box, click Enable.

### Configure the Back to top module

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Configuration > User interface > Back to Top.

## Back To Top ☆

Home » Administration » Configuration » User interface

- Prevent on mobile and touch devices  
Do you want to prevent Back To Top on touch devices?
- Prevent on administration pages and node edit  
Do you want to prevent Back To Top on admin pages?
- Prevent on front page  
Do you want to prevent Back To Top on front page?

### Trigger

Set the number of pixel which trigger the Back To Top button default 100

### Placement

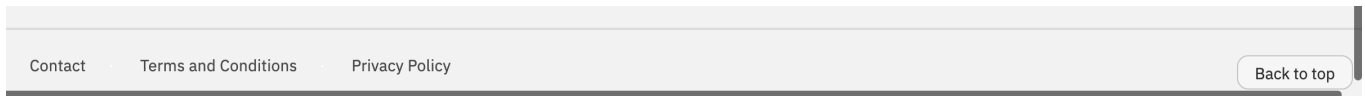
Where should the Back To Top button appear?

### Button text

Set the text of the Back To Top button

3. Change any settings as required, or keep the defaults.
4. Click Save configuration

You now have a Back to Top button.



## What you did in this tutorial

You added a Back to Top button, as a navigation enhancement to help your users get back to the top of a page that they are viewing, instead of scrolling.

## Tutorial: Creating a drop-down menu link

You can create and use drop-down menus, or nested menus, to enhance the use of your Developer Portal .

## Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial [Creating the Portal](#) explains how to enable the portal if you have not already done so.

## About this tutorial

In this tutorial, you create a drop-down menu link on the **Main navigation** menu.

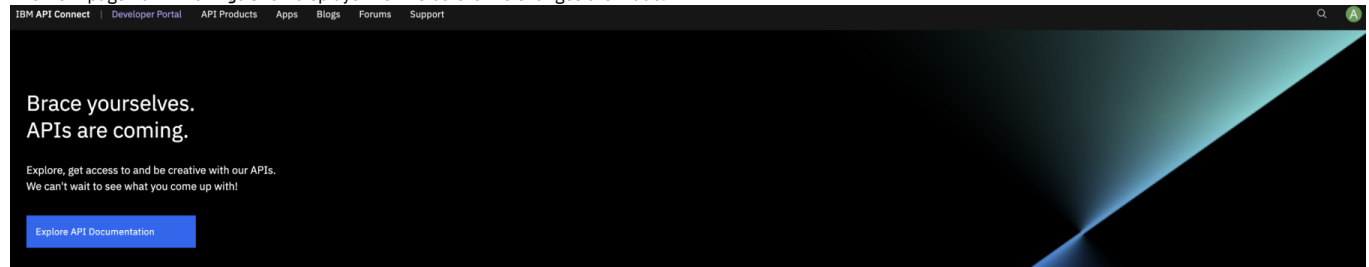
Note:

Although drop-down menu links can be created in the Developer portal, the drop-down menu doesn't expand by default. To make the drop-down menu expand, you need to use the **Superfish** module.

This tutorial takes you through the following steps:

1. [Add menu links to the main navigation menu.](#)
2. [Make the drop-down menu expand by using the Superfish module.](#)

The front page **Main navigation** displays like this before the changes are made.



## Add menu links to the main navigation menu

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.

3. Navigate to Structure > Menu > Main navigation > Add link
4. Enter API's in the **Menu link title**. Enter /api in the **Link**. Check Show as expanded, then click Save.  
Note: Ensure Show as expanded is checked as this is the parent link for the submenu links.
5. Add another menu link to the main navigation menu. Enter API One in the **Menu link title**. Enter /product/10/api/5 in the **Link**. Select -- API's in the Parent link, then click Save.

## Add menu link

[Home](#) » [Administration](#) » [Structure](#) » [Menus](#) » [Main navigation](#)

### Menu link title \*

The text to be used for this link in the menu.

### Link \*

- The location this menu link points to.
- Start typing the title of a piece of content to select it. You can also enter an internal path such as /node/add or an external URL such as http://example.com. Enter <frc

 Enabled

A flag for whether the link should be enabled in menus or hidden.

### Description

Shown when hovering over the menu link.

 Show as expanded

If selected and this menu link has children, the menu will always appear expanded. This option may be overridden for the entire menu tree when placing a menu block.

### Parent link

The maximum depth for a link and all its children is fixed. Some menu links may not be available as parents if selecting them would exceed this limit.

### Weight

Link weight among links in the same menu at the same depth. In the menu, the links with high weight will sink and links with a low weight will be positioned nearer the top.

6. Add another menu link to the main navigation menu. Enter API Two in the **Menu link title**. Enter /product/10/api/9 in the **Link**. Select -- API's in the Parent link. Select 1 in Weight, then click Save.

## Add menu link

[Home](#) » [Administration](#) » [Structure](#) » [Menus](#) » [Main navigation](#)

### Menu link title \*

The text to be used for this link in the menu.

### Link \*

- The location this menu link points to.
- Start typing the title of a piece of content to select it. You can also enter an internal path such as /node/add or an external URL such as http://example.com. Enter <from

 Enabled

A flag for whether the link should be enabled in menus or hidden.

### Description

Shown when hovering over the menu link.

 Show as expanded

If selected and this menu link has children, the menu will always appear expanded. This option may be overridden for the entire menu tree when placing a menu block.

### Parent link

The maximum depth for a link and all its children is fixed. Some menu links may not be available as parents if selecting them would exceed this limit.

### Weight

Link weight among links in the same menu at the same depth. In the menu, the links with high weight will sink and links with a low weight will be positioned nearer the top.

7. Navigate to Structure > Menus > Main navigation. The hierarchy of the menu links now looks like this:

**+ Add link**

**Title \***  
Main navigation Machine name: main

**Administrative summary**  
Site section links

**Menu language**  
English

MENU LINK	ENABLED	OPERATIONS
Home	<input checked="" type="checkbox"/>	Edit
API's	<input checked="" type="checkbox"/>	Edit
API One	<input checked="" type="checkbox"/>	Edit
API Two	<input checked="" type="checkbox"/>	Edit
API Products	<input checked="" type="checkbox"/>	Edit
Apps	<input checked="" type="checkbox"/>	Edit

8. Navigate to the home page. The **APIs** menu link that you added to the main navigation menu does appear, but the submenus links don't expand when you hover over it.



## Make the drop-down menu expand by using the Superfish module

To make the drop-down menu expand when you hover over it, you need to use the **Superfish** module and configure the **Block Layout**.

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Extend. Search for **superfish**, check the **Superfish** box, click Enable.

**+ Install new module**

superfish

Enter a part of the module name or description

Below is a list of installed modules. Modules that are i

**▼ USER INTERFACE**

**Superfish**

**Enable**

3. Navigate to Structure > Block layout.

4. Click Configure, and select Disable for the **Main menu** block.

BLOCK	CATEGORY	REGION	OPERATIONS
Navigation <span>Place block</span>			
Site branding	System	Navigation	Configure
Main menu	Menus	Navigation	Configure Disable Translate Remove Configure
Navigation Right <span>Place block</span>			
User menu	Menus	Navigation Right	

5. Click Place block for Navigation, Then search for `main navigation` in the box. The filtered results might show two or more options. Click Place block for the **Main navigation** with the Category **Superfish**.

**Place block**

[+ Add custom block](#)

BLOCK	CATEGORY	OPERATIONS
Main navigation	Menus	Place block
Main navigation	Superfish	Place block

6. Deselect **Display title**, click Save block.

**Configure block**

**Block description:** Main navigation

**Title \***

 Machine name: mainnavigation [\[Edit\]](#)

This field supports tokens. [Browse available tokens.](#)

Display title

[▶ MENU LEVELS](#)

**▼ BLOCK SETTINGS**

**Menu type**

Horizontal (single row)

Horizontal (double row)

Vertical (stack)

*(Default: Horizontal (single row))*

**Style**

None

*(Default: None)*

Add arrows to parent menus

Drop shadows

**Slide-in effect**

Vertical

*(Default: Vertical)*

The plugin provides a handful number of animation effects, they can be used by uploading the

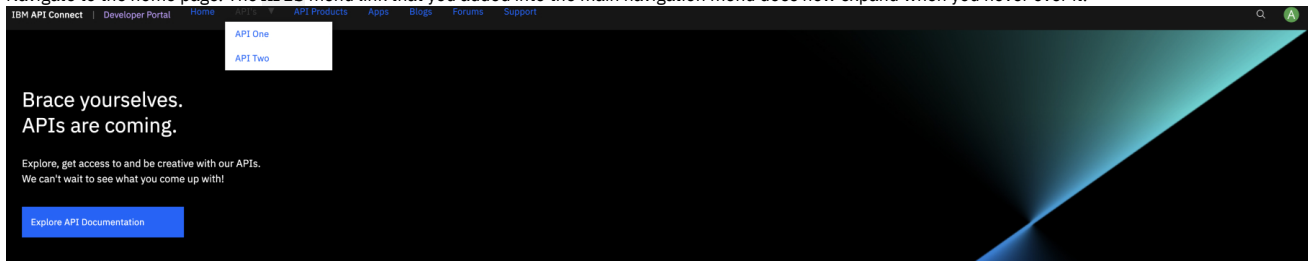
[Save block](#)

7. Drag **Main navigation** to be under **Site branding**.

BLOCK	CATEGORY	REGION	OPERATIONS
Navigation <span>Place block</span>			
Site branding	System	Navigation	Configure
Main navigation	Superfish	Navigation	Configure
Main menu (disabled)	Menus	Navigation	Enable

8. Click Save blocks.

9. Navigate to the home page. The **APIs** menu link that you added into the main navigation menu does now expand when you hover over it.



## What you did in this tutorial

You created a drop-down menu link on the **Main navigation** menu.

## What to do next

You can consider any other uses of the drop-down menu to enhance your users experience on your Developer Portal.

## Tutorial: Displaying blog posts on the front page

You can customize the appearance of the front page of your Developer Portal to display recent blog posts. You might use the blog posts to provide your users with information, such as service availability or marketing campaigns.

## Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial [Creating the Portal](#) explains how to enable the portal if you have not already done so.

## About this tutorial

In this task, you create a view, customize that view to show blog posts, then embed that view on the front page of your Developer Portal site.

Before you begin, the front page looks like this:



Let's get started!

Sign up

[Create a new account](#) and get started with our APIs. It's free to join.



Explore our APIs

Take a look at our [API products](#) and quickly find APIs to construct a fully featured application



Create

Select a plan and create your application to make use of our APIs.



This tutorial takes you through the following steps:

1. [Create a view.](#)
2. [Customize the view.](#)
3. [Embed the view on the front page of your Developer Portal.](#)

## Create a view

---

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Structure > Views > Add view.
4. In the **View name** block, enter a name for your view, for example Recent Blog posts.
5. Under **VIEW SETTINGS**, show **Content** of type **Blog Post** sorted by **Newest first**.
6. Check **Create a block** in **BLOCK SETTINGS**, and choose the number of blog posts that you want to be displayed.

**VIEW BASIC INFORMATION**

**View name \***

Recent Blog posts Machine name: recent\_blog\_posts [Edit]

Description

**VIEW SETTINGS**

Show: Content of type: Blog post sorted by: Newest first

**PAGE SETTINGS**

Create a page

**BLOCK SETTINGS**

Create a block

Block title

Recent Blog posts

**BLOCK DISPLAY SETTINGS**

Display format: Unformatted list of: titles (linked)

Items per block

3

Use a pager

[Save and edit](#) [Cancel](#)

7. Click Save and edit.

## Customize the view

---

You are now on the page where you can edit your view.

1. If you are not on the edit view page, navigate to Structure > Views, find the name of the view that you created, and click Edit.
2. Click Unformatted list in the **Format** section.
3. Select **Bootstrap List Group**, then click Apply.
4. Select Content: Title in the **Title field** drop-down, then click Apply.
5. Click Save.

## Embed the view on the front page of your Developer Portal

---

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Structure > Pages.
3. Click **Edit** on the welcome page.
4. Click **Panels**, then **Content**.
5. Click +Add new block.
6. Select your new block, which is under **LISTS (VIEWS)**.



Add block
✕

**Block description**  
Recent Blog posts

Display title

**Items per block**  
3

Override title

**Region \***  
Content

[Add block](#)

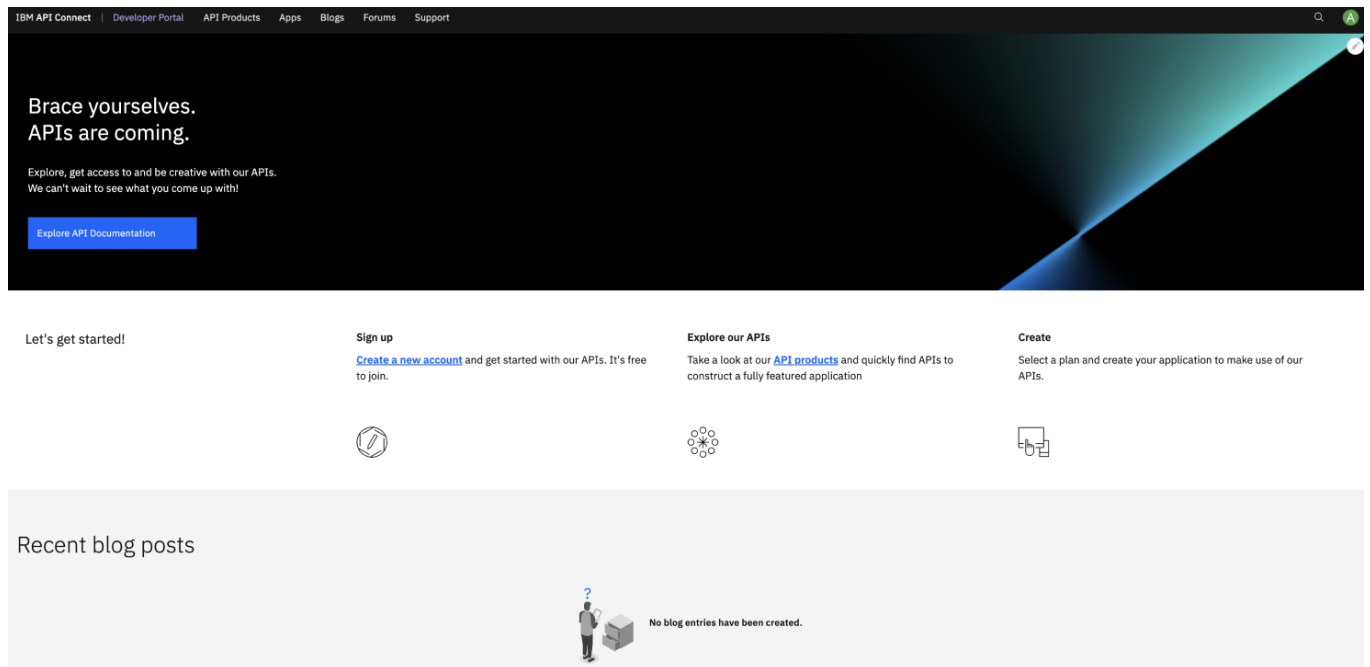
7. Select the number of **Items per block**, then click Add block.
8. You can now rearrange your view to be where you would like it on the front page.

LABEL	PLUGIN ID
<b>Content</b>	
✚ Welcome Banner	block_content:2fcd632-d4b9-44d9-b195-359e751966a1
✚ Featured Content	featuredcontent
✚ Recent Blog posts	views_block:recent_blog_posts-block_1
✚ Getting Started	block_content:f2b7762b-97f4-4d51-b7dc-d6de5be6d0ce

9. Click Update and save.

## What you did in this tutorial

You can check that the view you created appears on the front page of your site page.



## What to do next

You can edit your view at any time by navigating to Structure > Views and selecting your view by its name. You can also modify or delete the view's layout on the front page by navigating back to the layout section of your page under Structure > Pages.

## Tutorial: Displaying tweets on the front page

You can customize the appearance of the front page of your Developer Portal to display recent tweets. You might use the tweets to provide your users with information, such as service availability or marketing campaigns.

## Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial [Creating the Portal](#) explains how to enable the portal if you have not already done so.

## About this tutorial

In this task, you create a view, customize that view to show tweets, then embed that view on the front page of your Developer Portal site.

Before you begin, the front page looks like this:



Let's get started!

Sign up

[Create a new account](#) and get started with our APIs. It's free to join.



Explore our APIs

Take a look at our [API products](#) and quickly find APIs to construct a fully featured application



Create

Select a plan and create your application to make use of our APIs.



This tutorial takes you through the following steps:

1. [Enable the Media entity Twitter module.](#)
2. [Create an entity called Twitter to store the tweets.](#)
3. [Add media items.](#)
4. [Create a carousel view.](#)
5. [Customize the carousel view.](#)
6. [Embed the carousel view on the front page of your Developer Portal.](#)

## Enable the Media entity twitter module

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Extend, enter media in the search bar.
4. Check Media Entity Twitter, under MEDIA.

▼ MEDIA

<input checked="" type="checkbox"/>	<b>Enhanced Entity Browser</b>	► Provides some behavior and style enhancements to Entity Browsers, specifically for multiselect and image/media browsers.
<input checked="" type="checkbox"/>	<b>Media Entity Twitter</b>	► Media Entity Twitter provider.

5. Click Enable.

## Create an entity called Twitter to store the tweets

1. Navigate to Structure > Media types > Add media type.
2. In the Name field, enter Twitter.

Media source \*

Media source that is responsible for additional logic related to this media type.

3. In the Media source drop-down, select Twitter.
4. Ensure that Whether to use Twitter api to fetch tweets or not value is set to No.

5. Ensure the values in the FIELD MAPPING section are set to - Skip field -.

**Name \***  
 Machine name: twitter  
 The human-readable name of this media type.

**Description**  
 Describe this media type. The text will be displayed on the *Add new media* page.

**Media source \***  
  
 The media source cannot be changed after the media type is created.

**MEDIA SOURCE CONFIGURATION**  
 Tweet URL field is used to store the essential information about the media item.

**Whether to use Twitter api to fetch tweets or not.**  
  
 In order to use Twitter's api you have to create a developer account and an application. For more information consult the readme file.

Generate thumbnails  
 If checked, Drupal will automatically generate thumbnails for tweets that do not reference any external media. In certain circumstances

**FIELD MAPPING**  
 Media sources can provide metadata fields such as title, caption, size information, credits, etc. Media can automatically save this information.

**Tweet ID**

**Twitter user information**

6. Click Save.

7. Click Manage display from the Edit dropdown for the new `Twitter` media type.

8. Drag the Authored on, Authored by, and Thumbnail fields to the Disabled section, and set the FORMAT of the Tweet URL field to Twitter embed.

FIELD	LABEL	FORMAT
✚ Tweet URL	Above	Twitter embed
<b>Disabled</b>		
✚ Language	Above	Language
✚ Name	Above	Smart trimmed
✚ Search result excerpt		
✚ Authored on	- Hidden -	Default
✚ Authored by	- Hidden -	Author
✚ Thumbnail	- Hidden -	Image

▶ CUSTOM DISPLAY SETTINGS

[Save](#)

9. Click Save.

## Add media items

1. Navigate to Content > Media > Add media > Twitter.
2. In the Name field, enter a name for the media, for example Drupal Tweet.

3. In the Tweet URL field, enter a URL, for example, <https://twitter.com/drupal/status/966443708160839684>.

**Name \***

**Tweet URL \***

<b>Revision information</b> No revision	<b>Revision log message</b>  Briefly describe the changes you
<b>URL alias</b> No alias	
<b>Authoring information</b> By admin (1) on 2020-01-08	

Published

**Save**

4. Click Save.
5. Repeat these steps to add more tweets, you need at least 2 to make a carousel view.

## Create a carousel view

1. Navigate to Structure > Views > Add view.
2. In the View name block, enter a name for your view, for example Tweet view.
3. Under VIEW SETTINGS, show Media of type Twitter sorted by Unsorted.
4. In the BLOCK SETTINGS, section check Create a block and ensure it is Display format Slick Carousel of fields.

**VIEW BASIC INFORMATION**

**View name \***

 Machine name: tweet\_view [Edit]

Description

**VIEW SETTINGS**

Show:  of type:  sorted by:

**PAGE SETTINGS**

Create a page

**BLOCK SETTINGS**

Create a block

Block title

**BLOCK DISPLAY SETTINGS**

Display format:  of:

5. Click Save and edit.

## Customize the carousel view

You are now on the page where you can edit your view.

1. If you are not the edit view page, navigate to Structure > Views, find the name of the view that you created, and click Edit.
2. Click Tweet view and change it to the value <none>. Then, click Apply.

### Block: The title of this view

#### Title

This title will be displayed with the view, wherever titles are normally displayed; i.e., as the page title, block title, etc.

Apply

Cancel

3. Click Add in the FIELDS section, then search for tweets, select the check box for Tweet URL. Then, click Add and configure fields.



Tweet URL

Media

Appears in: twitter.

Selected: Tweet URL

Add and configure fields

Cancel

4. From the Formatter drop-down select Twitter embed, then click Apply.

### Configure field: Media: Tweet URL

Appears in: twitter.



Create a label



Exclude from display

Enable to load this field as hidden. Often used to group fields, or to use as token in another field.

#### Formatter

Twitter embed

▶ GLOBAL REPLACEMENT PATTERNS (FOR DESCRIPTION FIELD ONLY)

▶ STYLE SETTINGS

▶ REWRITE RESULTS

▶ NO RESULTS BEHAVIOR

▶ ADMINISTRATIVE TITLE

Apply

Cancel

[Remove](#)

5. In the FIELDS section, click Media: Name, then, click Remove.

**Configure field: Media: Name**

Create a label

Exclude from display  
Enable to load this field as hidden. Often used to group fields, or to use as token in another field.

**Formatter**

Plain text

Link to the Media

▶ GLOBAL REPLACEMENT PATTERNS (FOR DESCRIPTION FIELD ONLY)

▶ STYLE SETTINGS

▶ REWRITE RESULTS

▶ NO RESULTS BEHAVIOR

▶ ADMINISTRATIVE TITLE

[Apply](#) [Cancel](#) [Remove](#)

6. In the FILTER CRITERIA section, click the Media: Published (= True), then click Remove.

**Configure filter criterion: Media: Published**

Expose this filter to visitors, to allow them to change it

**Operator**

Is equal to

Is not equal to

**Published**

True

False

▶ GLOBAL REPLACEMENT PATTERNS (FOR DESCRIPTION FIELD ONLY)

▶ ADMINISTRATIVE TITLE

[Apply](#) [Cancel](#) [Remove](#)

7. In the FORMAT section, click Settings next to Slick Carousel.

8. Set Skin Main to Default.

9. In the CAPTION FIELDS section, check Media: Tweet URL, and Override main optionset.

10. In the OVERRIDABLE OPTIONS section, select Autoplay and Dots then, click Apply.

CAPTION FIELDS	
Media: Tweet URL <input checked="" type="checkbox"/>	
slick ID # <input type="text"/>	Cache <input type="text" value="&lt;No caching&gt;"/>
Override main optionset <input checked="" type="checkbox"/>	
OVERRIDABLE OPTIONS	
Arrows <input type="checkbox"/>	Autoplay <input checked="" type="checkbox"/>
Infinite <input type="checkbox"/>	Mousewheel <input type="checkbox"/>
	Dots <input checked="" type="checkbox"/>
	Variable width <input type="checkbox"/>
	Draggable <input type="checkbox"/>
	Randomize <input type="checkbox"/>

**Block: Pager options**

**Items to display**  
  
 Enter 0 for no limit.

**Offset (number of items to skip)**  
  
 For example, set this to 3 and the first 3 items will not be displayed.

11. In the PAGER section, clickItems, set the field to 0 then, click Apply.  
 12. Click Save.

## Embed the view on the front page of your Developer Portal

1. Navigate to Structure > Pages.
2. Click Edit on the welcome page.
3. Click Panels, then Content.
4. Click Add new block.

### LISTS (VIEWS)

- Active forum topics
- APIs
- Applications
- Blog
- New forum topics
- Product List
- Recent comments
- Recent content
- Tweet view
- Who's online
- Workbench: Current user: Overview block
- Workbench: Edits by user: Overview block
- Workbench: Recent content: Overview block

5. Select your new block, which is under LISTS(VIEWS).
6. Deselect Display title then, click Add block.

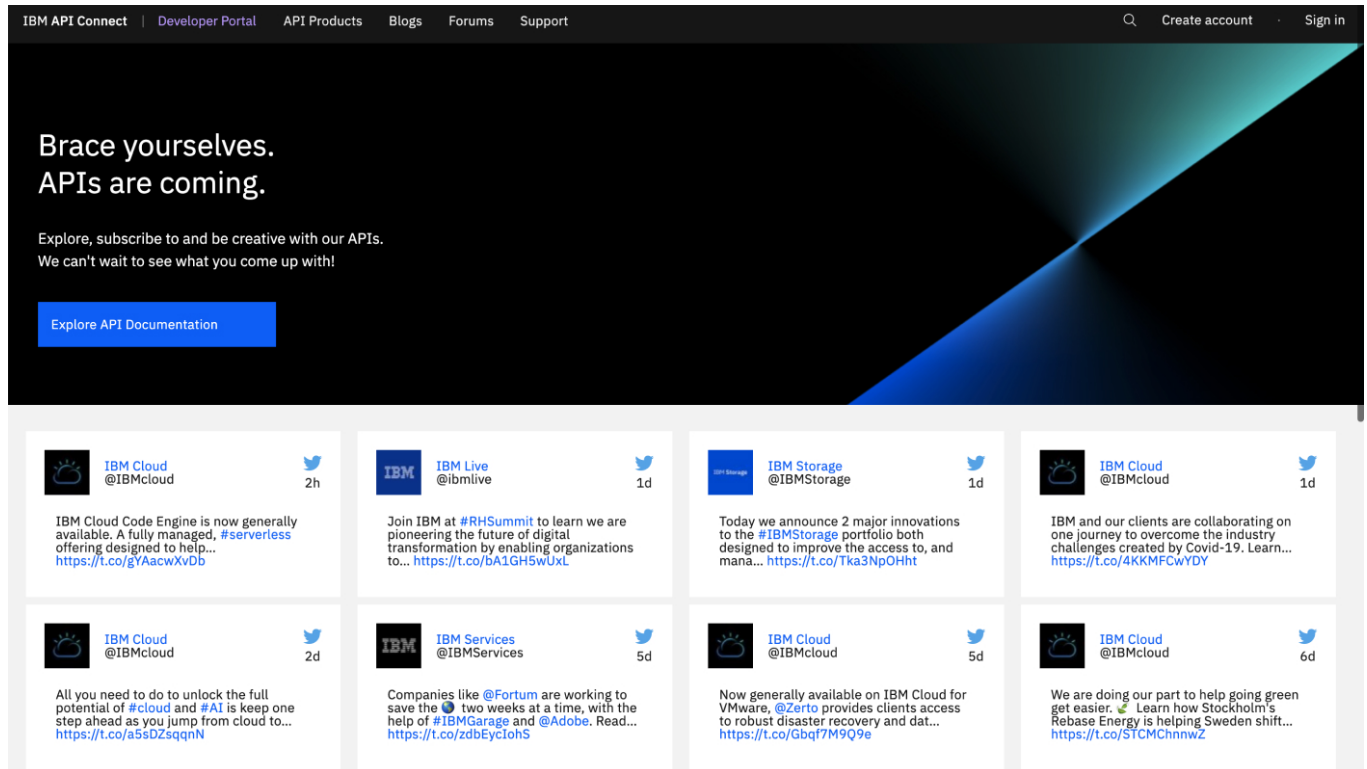
7. You can now rearrange your view to be where you would like it on the front page.

LABEL	PLUGIN ID
<b>Content</b>	
+	Welcome Banner block_content:d0f69e7a-e8c3-4f56-8537-13cea477f68a
+	Featured Content featuredcontent
+	Tweet view* views_block:tweet_view-block_1
+	Getting Started block_content:f56a934e-01ac-4083-8d80-da7169dd0289
+	Social Block social_block
+	Go ahead block_content:42d66e02-3835-4300-abc0-b719dcadd4e6
+	Get help block_content:777e45d7-90c4-499e-8727-e08215cae05a

8. Click Update and save.

## What you did in this tutorial

You can check that the view you created appears on the front page of your site page.



## What to do next

You can edit your view at any time by navigating to Structure > Views and selecting your view by its name. You can also modify or delete the view's layout on the front page by navigating back to the layout section of your page under Structure > Pages.

You can also integrate your Twitter data dynamically. For more information, see the links below.

## Related tasks

- [Integrating Twitter data into the social block](#)
- [Editing the social block](#)

## Tutorial: Changing the front page of your Developer Portal

You can create and arrange content in your Developer Portal to provide a front page that aligns with your brand, and helps your users discover your APIs.



## Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial [Creating the Portal](#) explains how to enable the portal if you have not already done so.

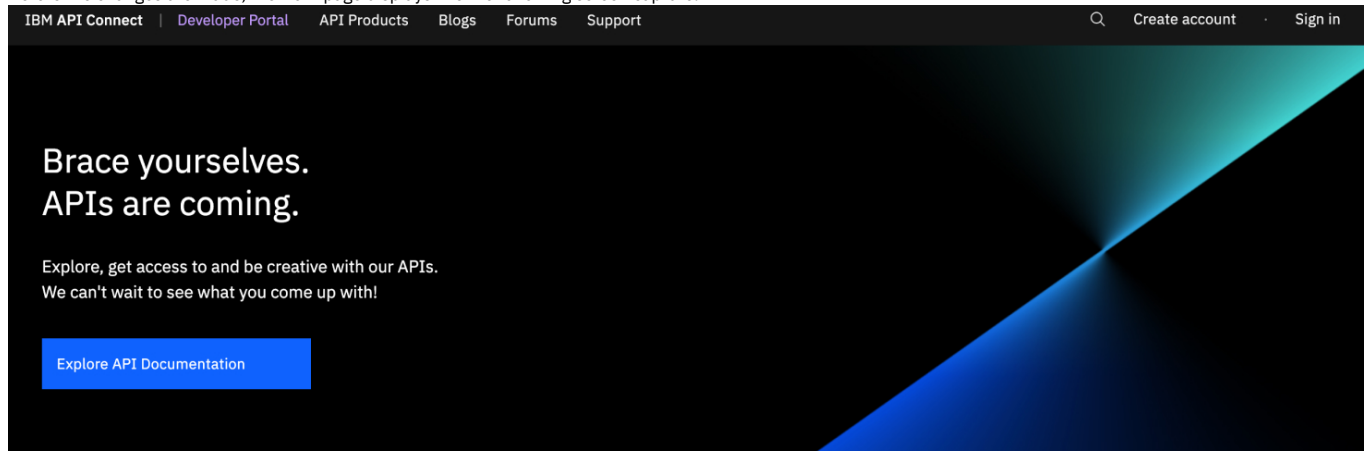
## About this tutorial

In this tutorial, you create some content for a page, and then change the front page to be your new page.

This tutorial takes you through the following steps:

1. [Creating some custom blocks](#).
2. [Create and configure a page](#).
3. [Adding the custom blocks to the page](#).
4. [Changing the front page to be your new page](#).

Before the changes are made, the front page displays like the following screen capture.



Let's get started!

Sign up

[Create a new account](#) and get started with our APIs. It's free to join.



Explore our APIs

Take a look at our [API products](#) and quickly find APIs to construct a fully featured application



Create

Select a plan and create your application to make use of our APIs.



## Creating some custom blocks

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Structure > Block layout > Add custom block
4. Enter `Block 1` in the Block description.
5. In the body area, click Source, then enter the following HTML:

```
<div class="tutorial_block_1 tutorial_block_frontpage">
<h1>Create with our APIs</h1>

<p>Welcome to our Developer Portal where you will find APIs to create your awesome app.</p>
</div>
```

## Add custom block ☆

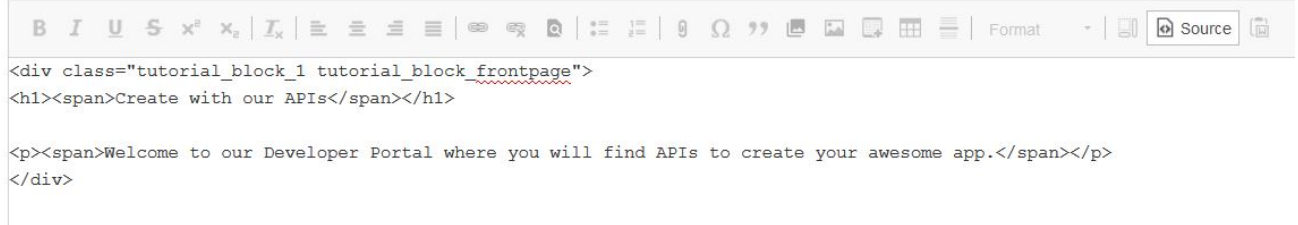
Home

### Block description \*

Block 1

A brief description of your block.

### Body



```
<div class="tutorial_block_1 tutorial_block_frontpage">
<h1>Create with our APIs</h1>

<p>Welcome to our Developer Portal where you will find APIs to create your awesome app.</p>
</div>
```

6. Check what the block looks like by clicking Source again.

## Add custom block ☆

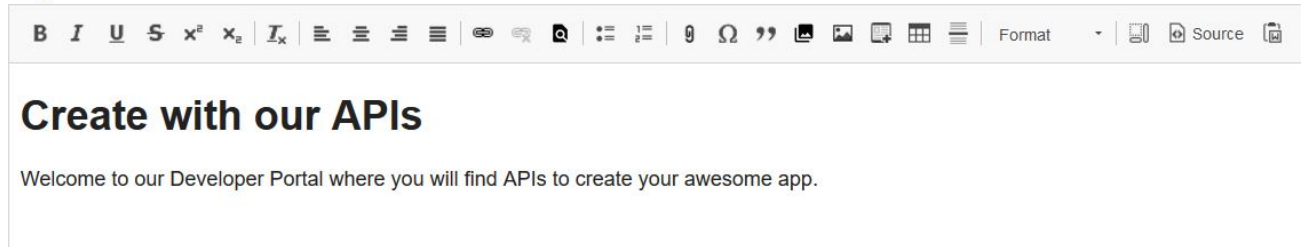
Home

### Block description \*

Block 1

A brief description of your block.

### Body



```
B I U S x² x₂ I_x | ≡ ≡ ≡ ≡ | 🔗 🗨 📄 | ⋮ ⋮ ⋮ ⋮ | 🔍 Ω ” 🖨 📷 📄 📄 📄 | Format - | 📄 Source 📄
```

# Create with our APIs

Welcome to our Developer Portal where you will find APIs to create your awesome app.

Note: This block uses no inline-styling, instead it allows your theme to decide how it is rendered. The advantage of this set up is that you can easily change the rendering without modifying existing content in the database. Any styling that is required can be added to a custom theme. For more information, see [Tutorial: Creating a custom theme for the Developer Portal](#).

7. Click Save.

You have created Block 1, repeat steps 3 to 7 to create Block 2, Block 3, and Block 4.

8. Enter Block 2 in the Block description. In the body area, click Source, then enter the following HTML:

```
<div class="tutorial_block_2 tutorial_block_frontpage">
<p class="text-align-center">Explore and subscribe to our APIs.

See what you come up with!</p>

<p class="text-align-center"> </p>

<div class="bannerButtonRow text-align-center"><a class="button cta-btns--white_btn"
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.devportal.doc_product">Explore API
Documentation</div>
</div>
```

9. Enter Block 3 in the Block description. In the body area, click Source, then enter the following HTML:

```
<div class="tutorial_block_3">
<p class="text-align-center">Create a new
account and get started with our APIs. It's free to join.</p>
</div>
```

10. Enter Block 4 in the Block description. In the body area, click Source, then enter the following HTML:

```
<div class="block-get-help">
<div class="get_help">
<div class="column col1">
<h3>Getting Help</h3>

<div>Be sure to check out these extra help resources.</div>
</div>

<div class="column col2">
<h4>Get Started</h4>

<div>Get started with our APIs by creating an account and exploring the documentation to find what's right for you.</div>
```

```

<div><svg
height="24"
id="_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.devportal.doc_tutorial_portal_example_frontpa
ge_Layer_1" space="preserve" version="1.1" viewBox="0 0 32 32" width="24" x="0px" xlink="http://www.w3.org/1999/xlink"
xmlns="http://www.w3.org/2000/svg" y="0px"> <g> <polygon points="15.293,10.707 19.586,15 8,15 8,17 19.586,17 15.293,21.293
16.707,22.707 23.414,16 16.707,9.293"></polygon> <path d="M16,2C8.269,2,2,8.269,2,16s6.269,14,14,14c7.731,0,14-6.269,14-
14S23.731,2,16,2z M16,28C9.383,28,4,22.617,4,16S9.383,4,16,4c6.617,0,12,5.383,12,12S22.617,28,16,28z"></path> </g>
</svg>Create an Account</div>
</div>

<div class="column col3">
<h4>Forum</h4>

<div>Ask a question in the forums or search to forum history to see if it's been asked before.</div>

<div><svg
height="24"
id="_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.devportal.doc_tutorial_portal_example_frontpa
ge_Layer_1" space="preserve" version="1.1" viewBox="0 0 32 32" width="24" x="0px" xlink="http://www.w3.org/1999/xlink"
xmlns="http://www.w3.org/2000/svg" y="0px"> <g> <polygon points="15.293,10.707 19.586,15 8,15 8,17 19.586,17 15.293,21.293
16.707,22.707 23.414,16 16.707,9.293"></polygon> <path d="M16,2C8.269,2,2,8.269,2,16s6.269,14,14,14c7.731,0,14-6.269,14-
14S23.731,2,16,2z M16,28C9.383,28,4,22.617,4,16S9.383,4,16,4c6.617,0,12,5.383,12,12S22.617,28,16,28z"></path> </g>
</svg>Join the discussion</div>
</div>

<div class="column col4">
<h4>Contact Us</h4>

<div>Can't find the answer to your question? Need more help? Have some feedback? Let us know!</div>

<div><svg
height="24"
id="_home_markdown_jenkins_workspace_Transform_in_SSMNED_10.0.5_com.ibm.apic.devportal.doc_tutorial_portal_example_frontpa
ge_Layer_1" space="preserve" version="1.1" viewBox="0 0 32 32" width="24" x="0px" xlink="http://www.w3.org/1999/xlink"
xmlns="http://www.w3.org/2000/svg" y="0px"> <g> <polygon points="15.293,10.707 19.586,15 8,15 8,17 19.586,17 15.293,21.293
16.707,22.707 23.414,16 16.707,9.293"></polygon> <path d="M16,2C8.269,2,2,8.269,2,16s6.269,14,14,14c7.731,0,14-6.269,14-
14S23.731,2,16,2z M16,28C9.383,28,4,22.617,4,16S9.383,4,16,4c6.617,0,12,5.383,12,12S22.617,28,16,28z"></path> </g>
</svg>Email us</div>
</div>
</div>
</div>

```

11. Navigate to Structure > Block layout > Custom block library to see the new blocks.

**Custom block library** ☆

Block layout Custom block library

Blocks Block types

Home » Administration » Structure » Block layout

Blocks in the block library belong to Custom block types, each with its own fields and display settings. After creating

**+ Add custom block**

Block description

Block type - Any -

**Apply**

BLOCK DESCRIPTION	BLOCK TYPE
Block 4	Basic block
Block 3	Basic block
Block 2	Basic block
Block 1	Basic block

## Create and configure a page

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Structure > Pages, click Add page.

[+ Add page](#)

LABEL	MACHINE NAME	PATH	OPERATIONS
Node view	node_view	/node/{node}	<a href="#">Edit</a> ▼
welcome	welcome	/home	<a href="#">Edit</a> ▼

3. Enter `Portal home` in the Administrative title field.
4. Enter `newhome` in the Path field.
5. Select **Panel1s** in the Variant title drop-down.
6. Leave other fields clear.
7. Click Next.
8. Accept the defaults of Panels for Labels, and Standard for Builder, click Next.

## Configure variant ☆

[Home](#) » [Administration](#) » [Structure](#) » [Pages](#) » [Add new page](#)

Page information » **Configure variant** » [Layout](#) » [Content](#)

### Label \*

Panels

### Builder

Standard ▼

[Previous](#)

[Next](#)

9. Select Two column bricks from Layout drop-down. Click Next.
10. Keep all settings except for the following:
  - Top Left, set Classes to Small:9
  - Top Right, set Classes to Small:3
  - Bottom Left, set Classes to Small:4
  - Bottom Right, set Classes to Small:8

### ▼ REGION: TOP LEFT

#### Wrapper

Div ▼

#### Classes

Small: 6 ▲  
Small: 7 ▲  
Small: 8 ▲  
**Small: 9** ▼

Add region specific classes: `bs-region` and `bs-region--top_left`

#### Additional attributes

E.g. `id|custom-id,role|navigation,data-something|some value`

[Browse available tokens.](#)

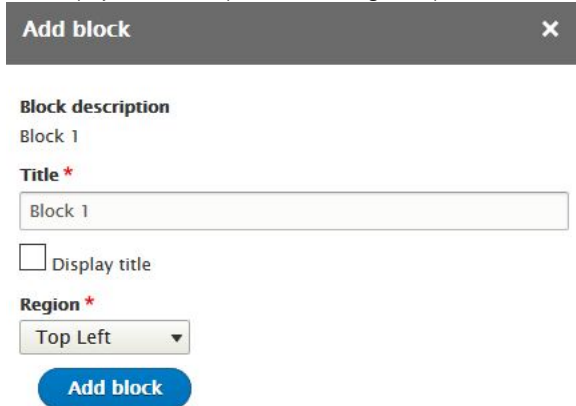
11. Click Next.

## Adding the custom blocks to the page

1. Leaving Page title blank, click Add new block.
2. Select Block 1 from the Custom block list.



3. Clear Display title, select Top Left from the Region drop down, then click Add block.



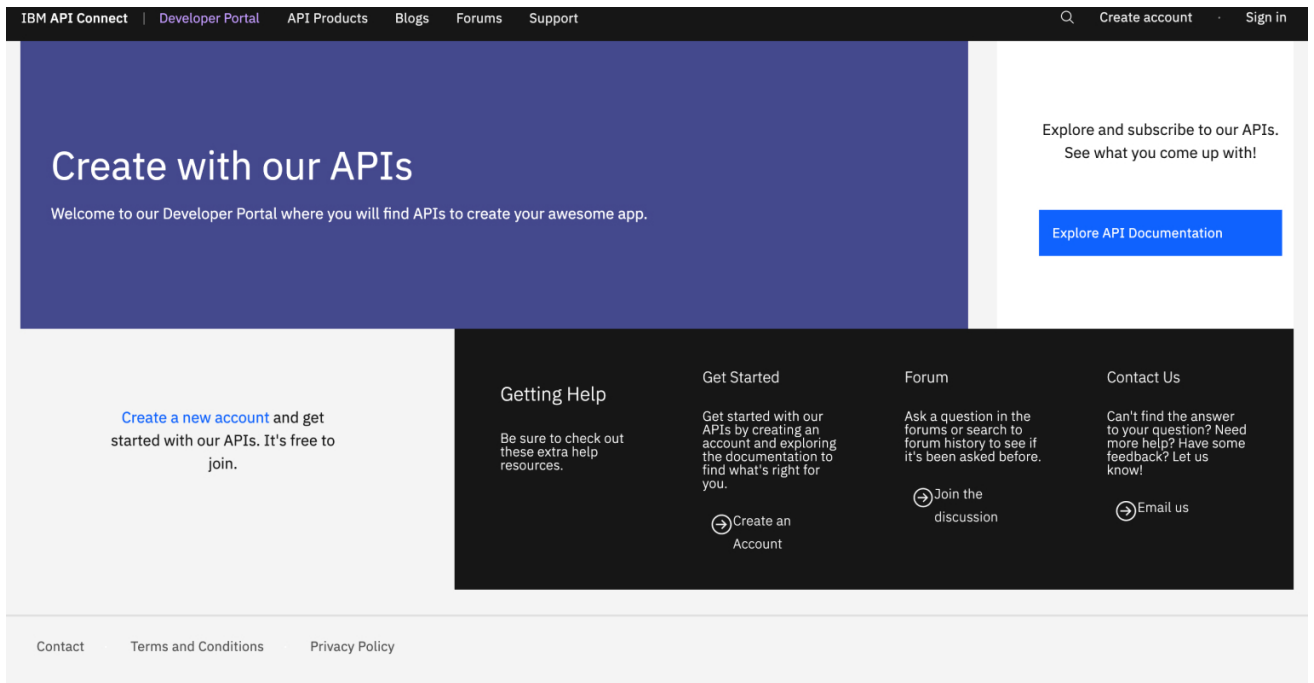
You added Block 1, repeat steps 1 to 3 to add Block 2, Block 3, and Block 4.

4. For Block 2 select region Top Right.
5. For Block 3 select region Bottom Left.
6. For Block 4 select region Bottom Right.
7. Click Finish.
8. Click Update and save.

## Changing the front page to be your new page

Note: In this tutorial, you create a new page and swap it with the default. If errors are made, the default page is still available as a backup.

1. Navigate to Structure > Pages.
2. Click the newhome URL link for your new Portal home page, check that your page looks as designed.



3. Navigate to Configuration > System > Basic site settings
4. In Front Page region set Default front page field tonewhome .
5. Click Save configuratoin.

## What you did in this tutorial

The front page of your Developer Portal is now changed and laid out with the blocks of content that you specified.

## What to do next

You can add further branding by adding content to your custom blocks such as images, or try to use one of the provided custom blocks such as the Featured Content Block. By default this block is English, you might want to consider adding versions in other languages.

## Tutorial: Grouping products by category

You can configure the Developer Portal so that products are displayed in groups according to the category that they are tagged with.

You can use the **Views** module to fetch content from your Developer Portal site, such as lists of products, and present the content to users in different formats. You can combine multiple views on a new page to display whatever content you want.

## Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial [Creating the Portal](#) explains how to enable the portal if you have not already done so.

## About this tutorial

In this tutorial you create three views in the Developer Portal, each one is configured to display a list of products according to their category. You then create a new page to show the views, where you see products that are grouped by category.

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Create three new taxonomy tags.
  - a. Navigate to Structure > Taxonomy > Tags.
  - b. Click **Add term**.
  - c. Enter *Accounts* in the **Name** field and click **Save**.
  - d. Enter *Logistics* in the **Name** field and click **Save**.
  - e. Enter *Marketing* in the **Name** field and click **Save**.
  - f. Navigate to Structure > Taxonomy > Tags to show the list of the new tags.

## Tags ☆

List Edit Manage fields Manage form display Manage display

Home » Administration » Structure » Taxonomy » Edit Tags

You can reorganize the terms in *Tags* using their drag-and-drop handles, and group terms under a parent term by sliding them under and to the right of the parent.

+ Add term

NAME	OPERATIONS
+ Accounts	Edit ▾
+ Logistics	Edit ▾
+ Marketing	Edit ▾

Save

Reset to alphabetical

4. Create a view to display products with the **Accounts** category.

- Navigate to Structure > Views > Add new view.
- Enter **Accounts products** in the **View name** field.
- In **View Settings**, select **Show: Content, of type: Product, sorted by: Newest first**, from the drop-down lists.
- In **Page settings** leave the **Create a page** cleared.
- In **Block settings** select **Create a block** and enter **Accounts products** in the **Block title**.
- Ensure **Block Display Settings** has **Display format: Masonry, of: titles (linked)**, and select **Use a pager**.

### PAGE SETTINGS

Create a page

### BLOCK SETTINGS

Create a block

Block title

Accounts products

### BLOCK DISPLAY SETTINGS

Display format: Masonry ▾ of: titles (linked) ▾

Items per block

5 ▾

Use a pager

Save and edit

Cancel

g. Click **Save and edit**.

5. Configure the view block settings.

- Under the **Format** section click the **Fields** link next to the **Show** label.
- In the dialog box, select the **Content** radio button then click **Apply**.
- Select **Card** from the **View mode** drop-down list, then click **Apply**.
- Click **Add** next to the **Filter Criteria** section.

- e. In the dialog box, search for `tag` then select the option **Tags** (`apic_tags`) from the list then click **Add and configure filter criteria**.
- f. In the next dialog box, select the **Tags** radio button and the **Client-side hierarchical** then click **Apply and continue**.
- g. Next set the **Operator** to **Is all of**, and from the **Select terms from vocabulary Tags** drop-down select **Accounts** and click **Apply**.
- h. Under the **Advanced** options, click the **None** link next to the **CSS class** label.
- i. If you are using the default API connect theme, then enter the CSS class name `product-view` and click **Apply**. If you have your own CSS class that you want to use, then enter its name in here instead.
- j. Click **Save** to save all the changes.

6. Create a view for the Logistics products.
  - a. Select **Structure > Views**.
  - b. Navigate to the **Accounts products** view and select the **Duplicate** option.
  - c. Give the duplicated view the name **Logistics products** and click **Duplicate**.
  - d. Click the Title **Accounts products** and change it to **Logistics products** and click **Apply**.
  - e. Under the **Filter Criteria** section click the **Content: Tags** (`=Accounts`) link.
  - f. Change the value in the **Select terms from vocabulary Tags** drop-down to **Logistics** and click **Apply**.
  - g. Click **Save** to save all the changes.
7. Repeat step 6 for Marketing products.
8. Next, create a page to display the views.
  - a. Select **Structure > Pages**.
  - b. Click **Add page**.
  - c. Enter **Products by category** in the **Administrative title**, and `products-by-category` in the **Path**. Select the **Panels** option from the **Variant type** drop-down list then click **Next**.

- d. Change the **Label** value to `Products grouped by category` and set the **Builder** to **Standard** then click **Next**.
- e. From the **Layout** drop-down list select the value **Three column** then click **Next**.
- f. Leave all the defaults on the next page and click **Next**.
- g. Click **+Add new block**.
- h. Select **Accounts products** from the **Lists (views)** section.



i. Ensure that **Display title** is selected and **Region** is set to **Left**, click **Add block**.

**Add block**
✕

**Block description**  
Accounts products

Display title

**Items per block**

Override title

**Region \***

**Add block**

j. Click **Add new block**. Select **Logistics products** view from the list and set the **Region** to **Middle**, click **Add block**.

k. Click **Add new block**. Select **Marketing products** view from the list and set the **Region** to **Right**, click **Add block**.

l. Enter **Products grouped by category** in the **Page title** field, and click **Finish**.

**Page title**

Configure the page title that will be used for this display.

**+ Add new block**

LABEL	PLUGIN ID	REGION
<b>Left</b>		
✚ Accounts products	views_block:accounts_products-block_1	Left ▾
<b>Middle</b>		
✚ Logistics products	views_block:logistics_products-block_1	Middle ▾
<b>Right</b>		
✚ Marketing products	views_block:marketing_products-block_1	Right ▾

9. From your API Manager, create three new products, or update existing ones, to give each one a category value of either Accounts, Logistics, or Marketing. Publish each of the products to the catalog with the configured Developer Portal.

10. Check that the **Products grouped by category** page shows your published products by category.

a. Select **Structure > Pages**.

b. Click the **/products-by-category** path link.

c. You are redirected to the **/products-by-category** page and can see the products that you assigned to these categories.

## What you did in this tutorial

You created three new views in the Developer Portal, and configured each one to display a list of products according to their category. You then created a new page to show the views.

**Accounts products**



Testproduct1 1.0.0  
★★★★★

---

Query Parameters 1.0.0

**Logistics products**



Testproduct2 1.0.0  
★★★★★

---

Query Parameters 1.0.0

**Marketing products**



Testproduct3 1.0.0  
★★★★★

---

Query Parameters 1.0.0

## Tutorial: Adding a field to the sign up form

You might want to collect and store additional information about your Developer Portal users. For example, you might want to collect a department name, employee ID, or similar data when a user registers for an account on the portal.

## Before you begin

---

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial [Creating the Portal](#) explains how to enable the portal if you have not already done so.

## About this tutorial

---

In this tutorial, you will add a Department code field to the Sign up form. This field is required for all portal users.

This tutorial takes you through the following steps:

1. [Adding a field to a user entity.](#)
2. [Adding the new field to the Sign up form.](#)
3. [Testing the new field.](#)

## Adding a field to a user entity

---

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click **Manage** to display it.
3. Navigate to **Configuration** > **People** > **Account settings** > **Manage fields**

You are on the **Manage fields** tab.

1. Click **+ Add field**.
2. In the **Select a field type** dropdown, select **Text (plain)**.
3. In the **Label** field enter, **Department code**.
4. Click **Save and continue**.

You are on the **Field settings** tab.

1. Accept the defaults of **Maximum length** (255) and **Allowed number of values** (Limited:1).
2. Click **Save field settings**.

You are on the **Edit** tab.

1. Add some help text. This is information that is displayed to the user to assist them completing the field on a form. Examples or links to further information are useful here. We are going to add an example in this tutorial.
2. Add **Example department code: DEP123** into the **Help text** field.
3. Check the **Required field**.
4. Click **Save settings**.

The field has now been created. You are back on the **Manage fields** tab.

1. Check that **Department code** is a listed field.

## Adding the new field to the Sign up form

---

1. If the administrator dashboard isn't displayed, click **Manage** to display it.
2. Navigate to **Configuration** > **People** > **Account settings** > **Manage form display**

You are on the **Manage form display** tab. These tabs, one for each form, show the fields that are displayed from the user entity, and in which order.

1. Switch to the **Register** tab. Notice that initially the new **Department code** field is underneath the **Disabled row** in the page. This means that it isn't shown on the form.

2. Click the Department code icon and drag it so that it is between Last name and Consumer organization in the list.

Settings Manage fields Manage form display Manage display Translate account settings

Default Register

Home » Administration » Configuration » People » Account settings » Manage form display

+ Add group

Show row weights

FIELD	WIDGET	Textfield size: 60	⚙
Status			
Username			
Roles			
Notify user about new account			
Current password			
E-mail address			
First Name	Textfield	Textfield size: 60	⚙
Last Name	Textfield	Textfield size: 60	⚙
Department code	Textfield	Textfield size: 60	⚙
Consumer organization	Textfield	Textfield size: 60	⚙
Password			
Disabled			
Language code	Language select		

3. Click Save.

## Testing the new field

1. Sign out as the Admin user.
2. Click **Create account**.
3. Observe that the **Department code** field is now part of the sign up form.

## What you did in this tutorial

You added a Department code field to the Sign up form, then tested the form to ensure that the **Department code** field is now part of the sign up form.

In reality, you would need to add this configuration in such a way that it is available for all sites. To achieve this you would include the configuration in your custom module. For more information, see [this Drupal documentation](#).

You might want to add extra validation to the Department code field on the Sign up form. For more information, see [Tutorial: Adding validation to a field on the sign-up form](#).

## Tutorial: Adding validation to a field on the sign-up form

You might want to add extra validation to a field to help your Developer Portal users. For example, you might want to validate a department name, employee ID, or similar data when a user registers for an account on the portal.

## Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. You must have completed the tutorial about [Adding a field to the sign up form](#), where you added a Department code field to the Sign-up form.

## About this tutorial

In this tutorial, you add some validation to the Department code field on the Sign-up form. This is a required field for all portal users and is a string of the format **DEPnnn**, where **DEP** is the string prefix and **nnn** is a three-digit number, for example, **DEP123**. The validation is done by using a custom module that is already available in [IBM APIC Portal - addons](#). However, for more information, see [Custom module development: creating a module skeleton](#).

This tutorial takes you through the following steps:

1. [Cloning and packaging the validation module](#).
2. [Installing and enabling the validation module](#).
3. [Testing the validation in the sign up form](#).

Look at the files we will use in this link [IBM APIC Portal - addons](#).

- `user_field_example.info.yml` - this is the metadata for the module, defining things like the name and version, and containing other important information that is required by Drupal to import and enable the module.

```
name: 'IBM APIC Portal - user field example'
type: module
description: 'IBM API Connect Developer Portal tutorial - Example of validating a custom user field'
package: 'Custom'
core_version_requirement: ^8 || ^9
version: 1.0.0
project: 'user_field_example'
dependencies:
 - ibm_apim
 - auth_apic
```

- `user_field_example.module` - this contains a validation function and adds this to the Sign-up form by using the `hook_form_alter` function.

```
<?php
use Drupal\Core\Form\FormStateInterface;

/**
 * Implementation of hook_form_alter() to alter the Sign up form
 */
function user_field_example_form_alter(&$form, FormStateInterface $form_state, $form_id) {
 if ($form_id === 'user_register_form') {
 $form['#validate'][] = 'user_field_example_validate_department_code';
 }
}

/**
 * Validate the Department code field on the Sign up form.
 *
 * Valid entry = DEPnnn
 * where n = single figure digit.
 */
function user_field_example_validate_department_code($form, &$form_state) {
 $dept_code = $form_state->getValue('field_department_code')[0]['value'];
 $valid = preg_match('/^DEP\d{3}$/i', $dept_code);
 if (!$valid) {
 $form_state->setErrorByName('field_department_code', t('Invalid department code.'));
 }
}
```

## Cloning and packaging the validation module

Validation is added to the `Department code` field on the Sign-up form by using a custom module.

1. Clone the custom module. Here is an example command if you are using Mac and Linux:

```
git clone https://github.com/ibm-apiconnect/devportal-addons
```

2. Package up the custom module.

```
cd devportal-addons/apic_v10/modules
tar -czf user_field_example.tgz user_field_example/
```

## Installing and enabling the validation module

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click **Manage** to display it.
3. Click **Extend > Install new module**.
4. Click **Choose file** from the **Upload a module or theme archive to install** section.
5. Browse to and select `user_field_example.tgz`, the packaged custom module that you created earlier. Click **Open**.
6. Click **Install**.
7. Click the **Enable newly added modules** link.
8. Enter `IBM APIC Portal - user field example` in the filter box.
9. Check the checkbox next to `IBM APIC Portal - user field example`.
10. Click **Enable**.

The module is now installed and enabled.

## Testing the validation in the sign-up form

1. Sign out as the Admin user.
2. Click **Create account**.
3. Complete all fields, set `Department code` to `ABC123`.
4. Click **Sign up**.

5. Observe the failure message and highlighting on the **Department code** field.

The screenshot shows a web form titled "API Developer Portal Sign up with nls Catalog User Registry". The form contains several input fields: Username (filled with "fred"), Email address (filled with "fred@apic.com"), First Name (filled with "Fred"), Last Name (filled with "Bloggs"), Department code (filled with "ABC123"), and Consumer organization (empty). A red error message "Invalid department code." is displayed below the Department code field. At the top of the page, a red notification bar states "1 error has been found: Department code".

6. Change the **Department code** field to **DEP245**, reenter the password and click **Sign up** to see that a valid code is accepted.

## What you did in this tutorial

You added some validation to the Department code field on the Sign-up form, then tested the form to ensure that the **Department code** field accepts only codes in the format of **DEPnnn**.

In reality, you would need to add this configuration in such a way that it is available for all sites. To achieve this availability, you would include the configuration in your custom module. For more information, see [Drupal custom modules](#).

## Tutorial: Adding a field group to group fields in a content type

You might want to change how your user account pages are displayed when you have lots of fields.

If you have many fields for one of your content types, for example, a user, then you might want to group some of these fields together when you display them to other users, or when these fields are being edited inside a user form.

Field groups allow you to create a parent field that contains other fields, set up as children, within that group. The **Fieldgroup** forms come with default HTML wrappers such as vertical tabs, horizontal tabs, accordions, fieldsets, or div wrappers.

## Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial.

## About this tutorial

You will create field groups for various groups of user information. These groups can then be seen under the users own section on the account view and edit pages.

This tutorial takes you through the following steps:

1. [Adding a field group called Consumer Organization to the user account display.](#)
2. [Adding a field group called Personal Details to the user form display.](#)

## Adding a field group called Consumer Organization to the user account display

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click **Manage** to display it.
3. Navigate to **Configuration > People > Account settings > Manage display**.
4. Click **+ Add field group**.

5. Select **tab** as your group type.
6. Enter **Consumer Organization** as the label.
7. Click **Save and continue**.
8. Set the default state to **open**. This sets the tab to be expanded by default.
9. Click **Create group**.

[Home](#) » [Administration](#) » [Configuration](#) » [People](#) » [Account settings](#) » [Manage display](#)

**Field group label**

**Default state**

**Description**

**ID**

**Extra CSS classes**

When the group is created, you can assign fields to that group and rearrange the group on the page. After you create the group, you are on the **Manage Display** page.

1. Drag your new group from the disable section up into the higher section, just underneath **Picture**.
2. Drag **URL**, **Consumer Organization**, and **Consumer Organization URL** underneath the **Consumer Organization** group row, and slightly to one side. It fixates itself underneath the parent, and this is now set as the child of that group.
3. Click **Save**.

FIELD	LABEL	FORMAT		
⊕ First Name	Above	Plain text		⚙
⊕ Last Name	Above	Plain text		⚙
⊕ Member for				
⊕ Picture	Above	Image	Original image	⚙
⊕ <b>Consumer Organization</b>		Tab	Tab: open <a href="#">delete</a>	⚙
⊕ URL	Above	Plain text		⚙
⊕ Consumer organization	Above	Plain text		⚙
⊕ Consumer organization URL	Above	Plain text		⚙
⊕ State	Above	Default		
⊕ Language code	Above	Language		⚙

You are now able to see the new group of fields on the account page of a user.

**Consumer Organization**

**URL**  
 /consumer-api/user-registries/574dbbea-e045-4457-91f3-2e8581a72d6c/users/e29beb15-2fd6-4583-854c-fe2854389ee7

**Consumer organization**  
 Joe Smith

**Consumer organization URL**  
 /consumer-api/orgs/7d9808be-eae5-44dc-a45c-551e17c2b7d1

## Adding a field group called **Personal Details** to the user form display

1. Navigate to Configuration » People » Account settings » Manage form display.

2. Click **Add field group**.
3. Select **tab** as your group type.
4. Enter **Personal Details** as the label.

[Home](#) » [Administration](#) » [Configuration](#) » [People](#) » [Account settings](#) » [Manage form display](#)

Add a new group \*

Tab ▼

Label \*

Personal Details Machine name: group\_personal\_details [\[Edit\]](#)

Save and continue

5. Click **Save and continue**.
6. Set the default state to **open**. This sets the tab to be expanded by default.
7. Click **Create group**.

Again, when the group is created, you can assign fields to that group and then rearrange the group on the page as you require. After you create the group, you are returned to the **Manage Display** page.

1. Drag your new group from the disabled section up into the higher section, onto the highest row.
2. Drag **First Name**, **Last Name**, **User picture upload**, and **Avatar Generator** to underneath the **Personal Details** group row. It fixates itself underneath the parent, and this is now set as the child of that group.
3. Click **Save**.

You and your user can now see the new grouped section when you edit their account.

Before:

The screenshot shows the user profile page for 'admin' in IBM API Connect. The page has a dark header with navigation links: IBM API Connect, Developer Portal, API Products, Apps, Blogs, Forums, and Support. The main content area is titled 'admin' and features a sidebar on the left with options: Edit, Shortcuts, Change Password, and View. The main profile area contains the following fields and information:

- Email address \***: sofa.ivanova1@ibm.com
- Status**: Active
- Roles**: Authenticated user, Administrator, Supersuser
- Your current logout threshold**: (empty field)
- First Name**: admin
- Last Name**: (empty field)
- Code Snippet Language**: - None -

Below the Code Snippet Language field, there is a note: "If you have a preferred programming language then you can set it here and it will be used as the default language for all API code snippets."

After:

## What you did in this tutorial

You have successfully created field groups for various groups of user information, and presented them to the user in its own section on the account view and edit pages.

You can check whether your field groups are applied, by viewing the profile of another user. Do this check by finding any other users published content and clicking the associated avatar. Or, from the administrator dashboard you can click People and then click a user. You can see the new Consumer Organization field group on their profile page. If you then click **Edit** for the user, you can see the Personal Details field group.

## What to do next

You can edit the user displays anytime by navigating back to Configuration > People > Account settings > Manage display or Manage form display.

You can delete the field group by clicking the **delete** button in the field group row on either of these pages. You can also modify the type of group it is, for example, vertical tabs, horizontal tabs, accordions, and field sets.

## Tutorial: Adding a custom block to the front page

You can customize the appearance of the front page of your Developer Portal by adding a custom block. Blocks are boxes of content that are rendered into an area, or region, of a web page such as the Products page or Apps page that can be displayed in regions, such as footer or sidebar, on your page.

## Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial [Creating the Portal](#) explains how to enable the portal if you have not already done so. The tutorial [Adding a custom block to a page other than the front page](#) explains how to add a custom block to any other page.

## About this tutorial

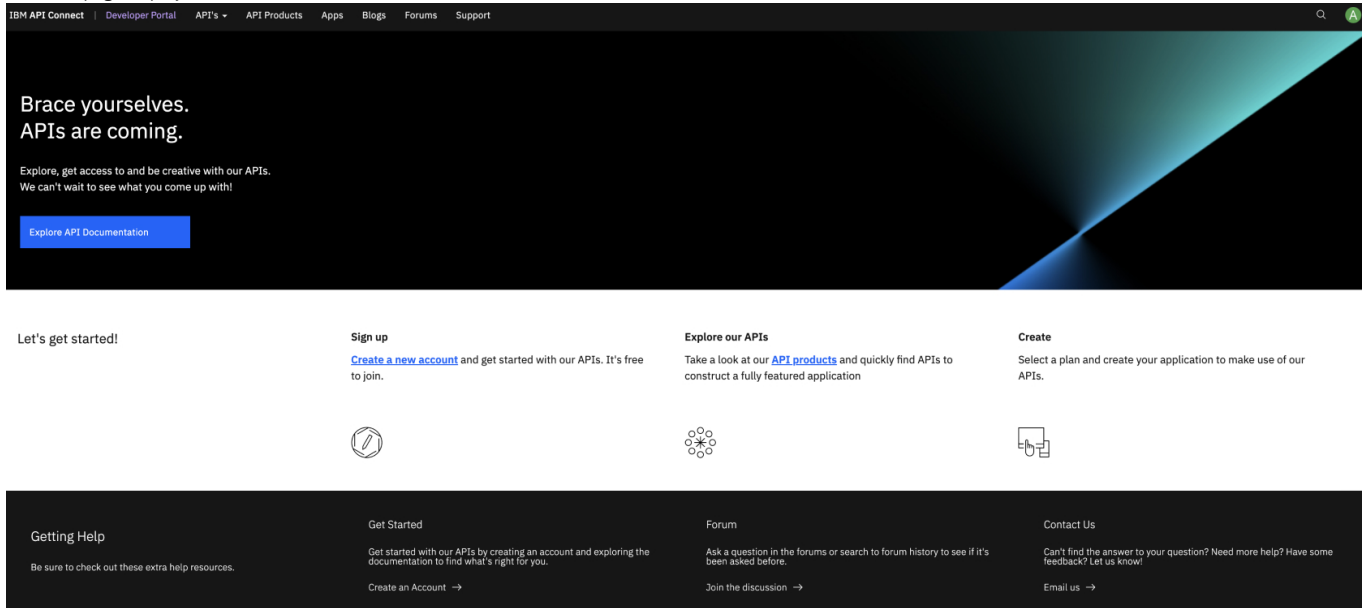
In this tutorial, you create a block that contains a marketing announcement and add it to the front page of your Developer Portal.

This tutorial takes you through the following steps:

1. [Creating a custom block](#).
2. [Adding the custom block to the front page](#).



The front page displays like this before the block is added.



## Creating a custom block

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click **Manage** to display it.
3. Navigate to **Structure > Block layout > Add custom block**.
4. Enter **Marketing Info** in the **Block description**.
5. In the body area, click **Source**, then enter this HTML:

```
<div class="tutorial_block_1 tutorial_block_frontpage">
<h1>Create with our APIs</h1>
<p>Welcome to our Developer Portal where you will find APIs to create your awesome app.</p>
</div>
```

### Add custom block ☆

Home

#### Block description \*

Marketing Info

A brief description of your block.

#### Body

Rich text editor toolbar with options for Bold, Italic, Underline, Strikethrough, Bulleted list, Numbered list, Indent, Outdent, Link, Unlink, Image, Video, Table, and more. A 'Source' button is visible on the right.

```
<div class="tutorial_block_1 tutorial_block_frontpage">
<h1>Create with our APIs</h1>

<p>Welcome to our Developer Portal where you will find APIs to create your awesome app.</p>
</div>
```

Text format: Full HTML

6. To see a preview of the block click **Source** again.

## Add custom block ☆

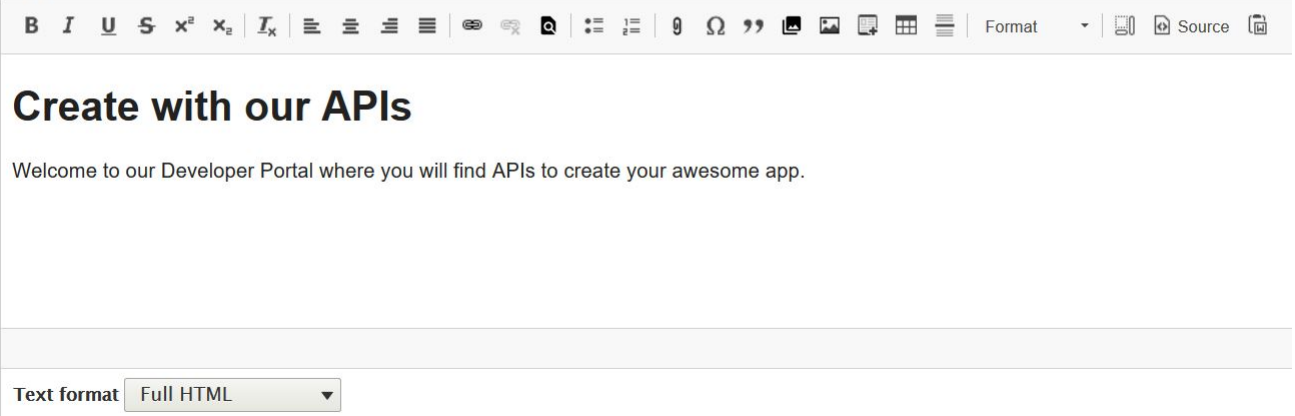
Home

### Block description \*

Marketing Info

A brief description of your block.

### Body



**B I U S x<sup>2</sup> x<sub>2</sub> I<sub>x</sub>** | [List] [List] [List] [List] | [Link] [Unlink] [Image] [Table] [Table] [Table] | Format [Source]

## Create with our APIs

Welcome to our Developer Portal where you will find APIs to create your awesome app.

Text format Full HTML

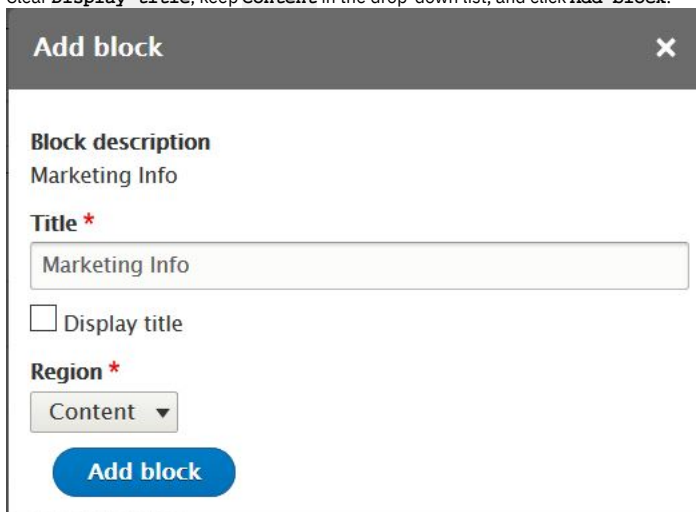
Note: This block uses almost no inline-styling, instead it allows your theme to decide how it is rendered. The advantage of this set up is that you can easily change the rendering without modifying existing content in the database. Any styling that is required can be added to a custom theme. For more information, see [Tutorial: Creating a custom theme for the Developer Portal](#).

7. Click **Save**.

You have now created the custom block.

## Adding the custom block to the front page

1. If the administrator dashboard isn't displayed, click **Manage** to display it.
2. Navigate to **Structure > Pages**.
3. Click **Edit** on the **Welcome** page label.
4. Click **Panels** in the **Variants** section.
5. Click **Content** on the expanded menu.
6. Click + **Add new block**.
7. Click **Marketing Info** in the Custom list.
8. Clear **Display title**, keep **Content** in the drop-down list, and click **Add block**.



**Add block** ✕

**Block description**  
Marketing Info

**Title \***  
Marketing Info

Display title

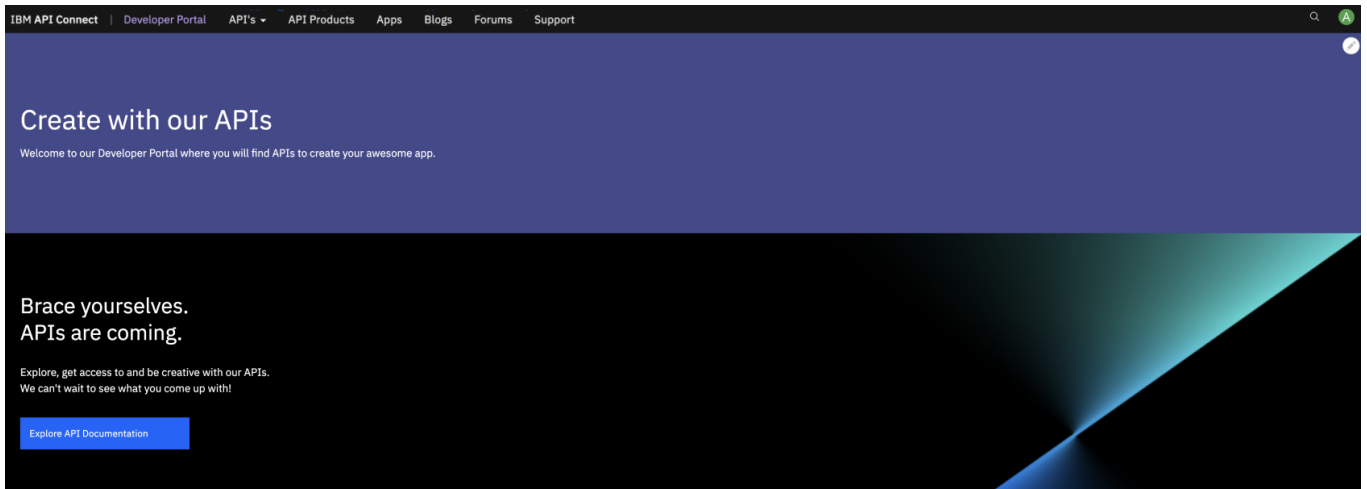
**Region \***  
Content

**Add block**

9. Drag **Marketing Info** to the order you want, in this example before the **Welcome banner** content.
10. Click **Update and save**.

## What you did in this tutorial

The front page now has the new block with a marketing message.



Let's get started!

Sign up

[Create a new account](#) and get started with our APIs. It's free to join.



Explore our APIs

Take a look at our [API products](#) and quickly find APIs to construct a fully featured application



Create

Select a plan and create your application to make use of our APIs.



## What to do next

You can add further branding by adding content to your custom blocks such as images, or try using one of the provided custom blocks such as the Featured Content Block. Instead of using inline styling like that provided in the html, adopt best practice by adding class selectors and developing a custom theme. By default this block is English, you might want to consider adding translated versions.

## Tutorial: Adding a custom block to a page other than the front page

You can customize the appearance of a page on your Developer Portal by adding a custom block. Blocks are boxes of content that are rendered into an area or region of a web page, such as the Products page or Apps page. Blocks can be displayed in regions, such as footer or sidebar, on your page.

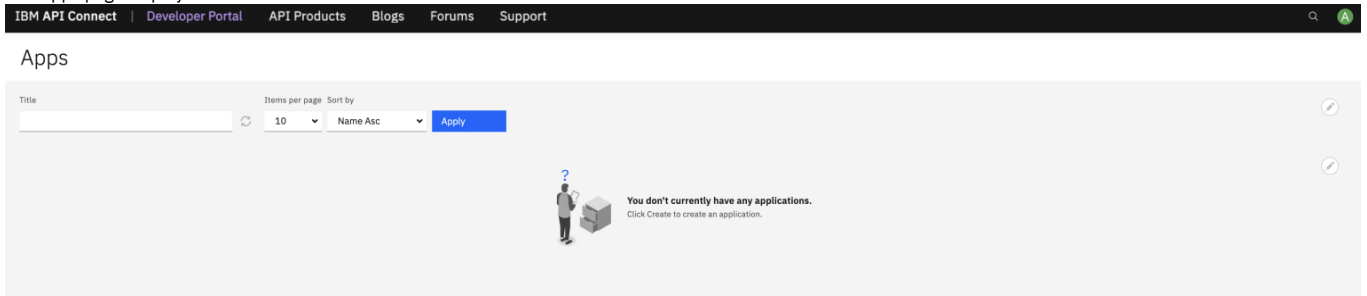
## Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial [Creating the Portal](#) explains how to enable the portal if you have not already done so. The tutorial [Adding a custom block to the front page](#) explains how to add a custom block to the front page.

## About this tutorial

In this tutorial, you create a custom block that contains extra guidance on application creation and registration, and add that custom block to the Apps page of your Developer Portal.

The Apps page displays like this before the block is added.



This tutorial takes you through creating a custom block, and configuring it to show on the Apps page.

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click **Manage** to display it.
3. Navigate to **Structure > Block layout > Add custom block**.
4. Enter **App Info** in the **Block description**.
5. In the body area, click **Source**, then enter this HTML:


```
<div class="tutorial_block_2">
<p>Before you can use an API, you must create an application. When you create an application you are provided with a client ID and client secret for the application. You must supply the client ID when you call an API that requires you to identify your application by using a client ID, or a client ID and client Secret.</p>
</div>
```

**Block description \***

App Info

A brief description of your block.

**Body**



```
<div class="tutorial_block_2">
<p>Before you can use an API, you must create an application. When you create an application you are provided with a client ID and client secret for the application. You must supply the client ID when you call an API that requires you to identify your application by using a client ID, or a client ID and client Secret.</p>
</div>
```

Text format: Full HTML [About text formats](#)


6. To see a preview of the block click **Source** again.

**Block description \***

App Info

A brief description of your block.

**Body**

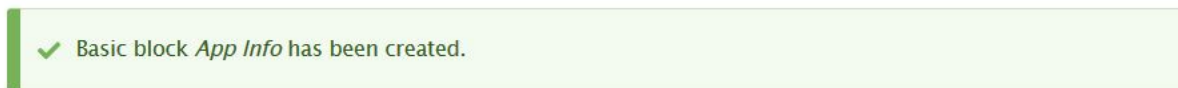


Before you can use an API, you must create an application. When you create an application you are provided with a client ID and client secret for the application. You must supply the client ID when you call an API that requires you to identify your application by using a client ID, or a client ID and client Secret.

Text format: Full HTML [About text formats](#)

Note: This block uses almost no inline-styling, instead it allows your theme to decide how it is rendered. The advantage of this set up is that you can easily change the rendering without modifying existing content in the database. Any styling that is required can be added to a custom theme. For more information, see [Tutorial: Creating a custom theme for the Developer Portal](#).

- Click **Save**.
- Enter **Creating an App** in the **Title** box, keep the **Display title** selected.



**Block description:** App Info

**Title \***

Creating an App

Machine name: appinfo [\[Edit\]](#)

This field supports tokens. [Browse available tokens.](#)

Display title

- Click **Pages** in the **Visibility** list.
- Enter **/application** in the **Pages** text area, keep the **Show for the listed pages** selected.
- Select **Secondary** from the **Region** drop-down menu.
- Click **Save block**.
- Click **Save blocks**.

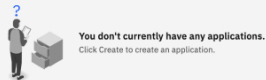
## What you did in this tutorial

The Apps page now has an additional block with a guidance message.

IBM API Connect | [Developer Portal](#) | [API Products](#) | [Blogs](#) | [Forums](#) | [Support](#)

Apps

Title:  Items per page: 10 Sort by: Name Asc [Apply](#)



**Creating an App**

Before you can use an API, you must create an application. When you create an application you are provided with a client ID and client secret for the application. You must supply the client ID when you call an API that requires you to identify your application by using a client ID, or a client ID and client Secret.

## What to do next

Instead of using inline styling like that provided in the html, adopt best practice by adding class selectors and developing a custom theme. By default this block is English, so you might want to consider adding translated versions.

## Tutorial: Configuring the RobotsTxt file

You can control the access of a visiting Web robot. You can configure the `robots.txt` file that exists on your web server, usually at the root level, to control access. Web robots are programs that crawl through the web to obtain web content for all the sites that are visited, and provide indexing for better performance of search engines. You can also specify separate rules for different robots.

### Why would I want to edit Drupal's pre-existing robots.txt file?

Malicious robots might choose not to honor the `robots.txt` file, and by editing this file you are broadcasting which sites you do not want others to see. Therefore, you should not use this file to hide sensitive data. Instead, you might want to edit your `robots.txt` file to:

- Prevent duplicate information from being identified on your site
- Prevent internal pages from appearing in search engines
- Prevent private pages from appearing in search engines
- Prevent particular images, files, and so on, from being crawled
- Specify a `crawl-delay` attribute to prevent robots from overloading your server at load time
- Exclude a particular robot

### Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial.

### About this tutorial

You will edit the pre-existing `robots.txt` file and exclude access to a visiting robot called BadBot.

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click **Manage** to display it.
3. Navigate to **Configuration > Search and Metadata > RobotsTxt**.

#### RobotsTxt

[Home](#) » [Administration](#) » [Configuration](#) » [Search and metadata](#)

See <http://www.robotstxt.org/> for more information concerning how to write your `robots.txt` file.

#### Contents of robots.txt

```
#
robots.txt
#
This file is to prevent the crawling and indexing of certain parts
of your site by web crawlers and spiders run by sites like Yahoo!
and Google. By telling these "robots" where not to go on your site,
you save bandwidth and server resources.
#
This file will be ignored unless it is at the root of your host:
Used: http://example.com/robots.txt
Ignored: http://example.com/site/robots.txt
#
For more information about the robots.txt standard, see:
http://www.robotstxt.org/robotstxt.html

User-agent: *
CSS, JS, Images
Allow: /core/*.css$
Allow: /core/*.css?
Allow: /core/*.js$
Allow: /core/*.js?
```

**Save configuration**

4. Enter the policy to exclude access to a robot called BadBot.

```
User-agent: BadBot
Disallow: /
```

5. Click **Save Configuration** to save your changes.

## What you did in this tutorial

---

You have now successfully customized the `robots.txt` file. Robots now use this updated file to decide where they can crawl on your site. The BadBot robot is excluded access.

You can check whether your `robots.txt` file is successfully changed by navigating to your site and appending `/robots.txt`. You should see the content you entered into that file.

```
#
robots.txt
#
This file is to prevent the crawling and indexing of certain parts
of your site by web crawlers and spiders run by sites like Yahoo!
and Google. By telling these "robots" where not to go on your site,
you save bandwidth and server resources.
#
This file will be ignored unless it is at the root of your host:
Used: http://example.com/robots.txt
Ignored: http://example.com/site/robots.txt
#
For more information about the robots.txt standard, see:
http://www.robotstxt.org/robotstxt.html
```

```
User-agent: BadBot
Disallow: /
```

```
User-agent: *
CSS, JS, Images
Allow: /core/*.css$
Allow: /core/*.css?
Allow: /core/*.js$
Allow: /core/*.js?
```

For more information on how to edit your `robots.txt` file, see <https://www.robotstxt.org/>.

## What to do next

---

You can edit the `robots.txt` at any time by navigating back to the page within the configuration settings. You might choose to duplicate this file across all of your sites, or choose different policies for different sites.

## Tutorial: Configuring the SecurityTxt file

---

You can use the `security.txt` file to provide your users with a standard way to report security vulnerabilities with your site.

The `security.txt` file is a file format that is designed to help your users disclose any security vulnerability. For more information, see [securitytxt.org](https://securitytxt.org) or [A File Format to Aid in Security Vulnerability Disclosure](#).

## Before you begin

---

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial.

## About this tutorial

---

You edit the `security.txt` file and provide information such as a URL of a contact page, for your customers to use if they have a security vulnerability.

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Configuration > System > Security.txt.

Back to site Workbench Manage Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

## Security.txt configure

Configure

Home » Administration » Configuration » System

A security.txt file provides a standard way for people to find out how to report security issues with your site. The new [Security.txt standard](#) is currently a [draft RFC](#).

Enable the security.txt file for your site

When enabled the security.txt file will be accessible to all users with the "view securitytxt" permission, you will almost certainly want to give this permission to everyone i.e. authenticated and anonymous users.

**CONTACT**

You must provide at least one means of contact: email, phone or contact page URL.

**Email**

Typically this would be of the form `security@example.com`. Leave it blank if you do not want to provide an email address.

**Phone**

Use full international format e.g. `+1-201-555-0123`. Leave it blank if you do not want to provide a phone number.

**URL**

The URL of a contact page which should be loaded over HTTPS. Leave it blank if you do not want to provide a contact page.

**ENCRYPTION**

Allow people to send you encrypted messages by providing your public key.

**Public key URL**

The URL of page which contains your public key. The page should be loaded over HTTPS.

**POLICY**

A security and/or disclosure policy can help security researchers understand what you are looking for and how to report security vulnerabilities.

**Security policy URL**

The URL of a page which provides details of your security and/or disclosure policy. Leave it blank if you do not have such a page.

**ACKNOWLEDGEMENT**

A security acknowledgements page should list the individuals or companies that have disclosed security vulnerabilities and worked with you to fix them.

**Acknowledgements page URL**

The URL of your security acknowledgements page. Leave it blank if you do not have such a page.

Save configuration

4. Select the checkbox for **Enable the security text file for your site**.
5. Complete the form for your requirements.
6. Click **Save Configuration** to save your changes.

## What you did in this tutorial

You successfully customized the `security.txt` file.

## What to do next

You can edit the `security.txt` at any time by navigating back to the page within the configuration settings.

## Tutorials for using custom modules in the Developer Portal

You can configure custom modules to complete specific actions in the Developer Portal.

Note: You must have administrator access to complete the tutorials.

You can configure rules for the following events that occur in the Developer Portal:

- Change the profile page to display **firstname last name**, instead of **username**.
- Add validation to a field on the sign up form.

---

# Tutorial: Changing the profile page to display firstname lastname, instead of username

You might want to change the profile page of your users on your Developer Portal to display their `firstname` and `lastname` rather than their `username`.

## Before you begin

---

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial.

## About this tutorial

---

In this tutorial, you change the profile page of your users to display their `firstname` and `lastname` rather than their `username`. The change to the profile page is done by using the custom module `change_account_display` that is already available in [IBM APIC Portal - addons](#). However, for more information, see [Custom module development: creating a module skeleton](#).

This tutorial takes you through the following steps:

1. [Cloning and packaging the `change\_account\_display` module](#).
2. [Installing and enabling the `change\_account\_display` module](#).

## Cloning and packaging the `change_account_display` module

---

1. Clone the custom module. Here is an example command if you are using Mac and Linux:

```
git clone https://github.com/ibm-apiconnect/devportal-addons
```

2. Package up the custom module.

```
cd devportal-addons/apic_v10/modules
tar -czf change_account_display.tar.gz change_account_display/
```

## Installing and enabling the `change_account_display` module

---

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click **Manage** to display it.
3. Click **Extend** > **Install new module**.
4. Click **Browse** from the **Upload a module or theme archive to install** section.
5. Navigate to and select the packaged custom module that you created earlier. For example, `change_account_display.tar.gz`, then click **Open**.

# API Developer Portal

## Update manager

✓ Installation was completed successfully.

### change\_account\_display

- Installed *change\_account\_display* successfully

### Next steps

- [Install another module](#)
- [Enable newly added modules](#)
- [Administration pages](#)

6. Click **Install**.
7. Click the **Enable newly added modules** link.
8. Enter **Change Account Display** in the filter box.
9. Check the check box next to **Change Account Display**.
10. Click **Enable**.

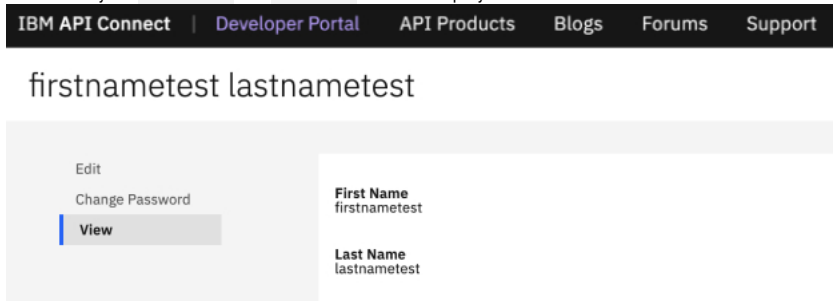
The module is now installed and enabled.

## Test the profile page change

---



1. Sign out as the Admin user.
2. Sign in as a user of your Developer Portal.
3. Click **My account** from the drop down next to your **username**.
4. Notice that your **firstname** and **lastname** are now displayed.



## What you did in this tutorial

You used a custom module to change the profile page display for the users of your Developer Portal.

## Tutorial: Adding validation to a field on the sign-up form

You might want to add extra validation to a field to help your Developer Portal users. For example, you might want to validate a department name, employee ID, or similar data when a user registers for an account on the portal.

## Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. You must have completed the tutorial about [Adding a field to the sign up form](#), where you added a Department code field to the Sign-up form.

## About this tutorial

In this tutorial, you add some validation to the Department code field on the Sign-up form. This is a required field for all portal users and is a string of the format **DEPnnn**, where **DEP** is the string prefix and **nnn** is a three-digit number, for example, **DEP123**. The validation is done by using a custom module that is already available in [IBM APIC Portal - addons](#). However, for more information, see [Custom module development: creating a module skeleton](#).

This tutorial takes you through the following steps:

1. [Cloning and packaging the validation module](#).
2. [Installing and enabling the validation module](#).
3. [Testing the validation in the sign up form](#).

Look at the files we will use in this link [IBM APIC Portal - addons](#).

- `user_field_example.info.yml` - this is the metadata for the module, defining things like the name and version, and containing other important information that is required by Drupal to import and enable the module.

```
name: 'IBM APIC Portal - user field example'
type: module
description: 'IBM API Connect Developer Portal tutorial - Example of validating a custom user field'
package: 'Custom'
core_version_requirement: ^8 || ^9
version: 1.0.0
project: 'user_field_example'
dependencies:
 - ibm_apim
 - auth_apic
```

- `user_field_example.module` - this contains a validation function and adds this to the Sign-up form by using the `hook_form_alter` function.

```
<?php
use Drupal\Core\Form\FormStateInterface;

/**
 * Implementation of hook_form_alter() to alter the Sign up form
 */
function user_field_example_form_alter(&$form, FormStateInterface $form_state, $form_id) {
 if ($form_id === 'user_register_form') {
 $form['#validate'][] = 'user_field_example_validate_department_code';
 }
}

/**
 * Validate the Department code field on the Sign up form.
 *
 * Valid entry = DEPnnn
 * where n = single figure digit.
 */
function user_field_example_validate_department_code($form, &$form_state) {
 $dept_code = $form_state->getValue('field_department_code')[0]['value'];
```

```

 $valid = preg_match('/^DEP\d{3}$/', $dept_code);
 if (!$valid) {
 $form_state->setErrorByName('field_department_code', t('Invalid department code.));
 }
}

```

## Cloning and packaging the validation module

Validation is added to the `Department code` field on the Sign-up form by using a custom module.

1. Clone the custom module. Here is an example command if you are using Mac and Linux:

```
git clone https://github.com/ibm-apiconnect/devportal-addons
```

2. Package up the custom module.

```
cd devportal-addons/apic_v10/modules
tar -czf user_field_example.tgz user_field_example/
```

## Installing and enabling the validation module

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click **Manage** to display it.
3. Click **Extend**, **Install new module**.
4. Click **Choose file** from the **Upload a module or theme archive to install** section.
5. Browse to and select `user_field_example.tgz`, the packaged custom module that you created earlier. Click **Open**.
6. Click **Install**.
7. Click the **Enable newly added modules** link.
8. Enter **IBM APIC Portal - user field example** in the filter box.
9. Check the checkbox next to **IBM APIC Portal - user field example**.
10. Click **Enable**.

The module is now installed and enabled.

## Testing the validation in the sign-up form

1. Sign out as the Admin user.
2. Click **Create account**.
3. Complete all fields, set **Department code** to `ABC123`.
4. Click **Sign up**.
5. Observe the failure message and highlighting on the **Department code** field.

❗ 1 error has been found: **Department code** ✕

API Developer Portal

## Sign up

Sign up with nls Catalog  
User Registry

**Username \***

**Email address \***

**First Name \***

**Last Name \***

**Department code \***

Invalid department code.

**Consumer organization \***

6. Change the **Department code** field to `DEP245`, reenter the password and click **Sign up** to see that a valid code is accepted.

## What you did in this tutorial

You added some validation to the Department code field on the Sign-up form, then tested the form to ensure that the **Department code** field accepts only codes in the format of **DEPnnn**.

In reality, you would need to add this configuration in such a way that it is available for all sites. To achieve this availability, you would include the configuration in your custom module. For more information, see [Drupal custom modules](#).

## Tutorial: Using a custom weighting sort order on the product list page

You can use a custom weighting to sort your products on the API products list page of your Developer Portal.

### Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial [Creating the Portal](#) explains how to enable the portal if you have not already done so.

### About this tutorial

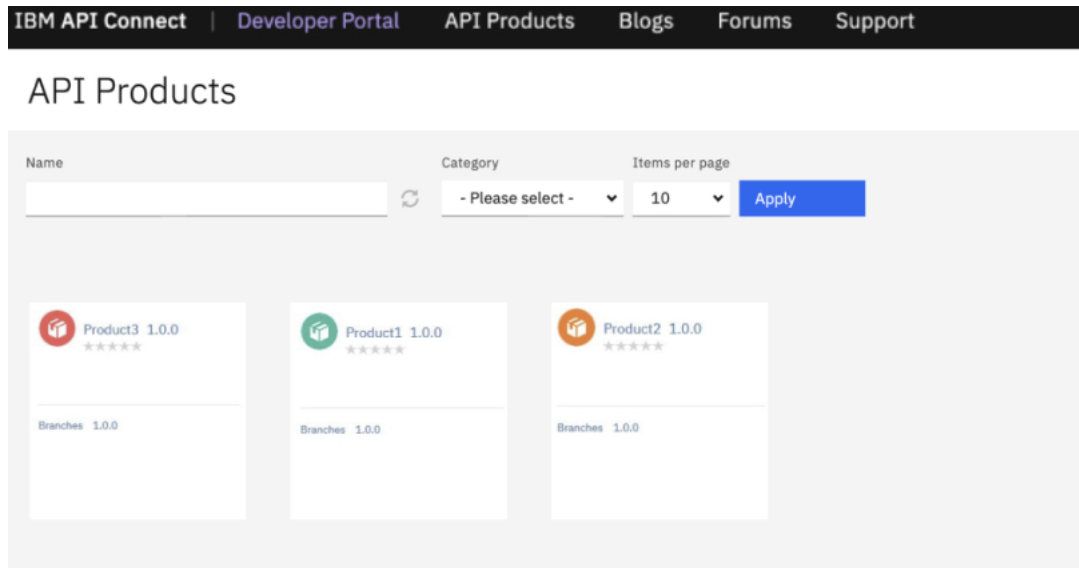
In this task, you set up a weighting sort criteria and use it to list your products, on the API Products page, in a particular order. You set up the weightings so the order reads, Product3, Product1, and Product2.

This tutorial takes you through the following steps:

1. [Add a custom integer field to the API content type.](#)
2. [Modify the Products view to change the sort order.](#)
3. [Add a weighting to each Product.](#)

Note: To complete this tutorial, you will need to have API products.

For example, the API Products page below displays Product1, Product2, and Product3 in this order before the weightings are added.



### Add a custom integer field to the API content type

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Structure > Content types > Product > Manage fields.
4. Click + Add field.
5. In **Add a new field**, select Number (Integer). Enter **Weighting** as the label title.

## Add field ☆

Home » Administration » Structure » Content types » Product » Manage fields

### Add a new field

Number (integer) ▼

OR

### Re-use an existing field

- Select an existing field - ▼

### Label \*

Weighting

Machine name: field\_weighting [Edit]

Save and continue

6. Click Save and continue.
7. Leave the **Allowed number of values** set to 1 and click Save field settings.
8. Enter 0 as the default value, click Save settings.
9. Click Save.

The **Weighting** field now shows in the **Manage fields** list.

## Modify the Products view to change the sort order

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Structure » Views, scroll down to the **Products** view and click Edit.
3. In the **SORT CRITERIA** section click Add.
4. Enter **weight** in the search box. Then, select **Weighting (field\_weighting)**, and click Apply (all displays).
5. Leave the **Order** as **Sort ascending**, click Apply (all displays).
6. In the **SORTCRITERIA** section, select **Rearrange** from the drop-down menu next to the Add.

### Rearrange sort criteria

For: All displays (except overridden) ▼

Show row weights

	REMOVE
Content: Title Exposed	Remove
Content: Version (apic_version) Exposed	Remove
Content: Changed Exposed	Remove
Content: Authored on Exposed	Remove
Content: Rating (apic_rating) Exposed	Remove
Content: Weighting (field_weighting) asc	Remove

Apply (all displays) Cancel

7. For all rows except **Weighting (field\_weighting)**, click **Remove**.
8. Click **Apply (all displays)**.

You modified the **Products** view to sort based on **Weighting (field\_weighting)**.

## Add a weighting to each Product

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Click Content, and select **Product** from the **Content type**, click Filter. A list of available Products on your site is displayed.

+ Add content

Title:  Content type: Product Published status: - Any - Language: - Any -

Filter

Action: Delete content

Apply to selected items

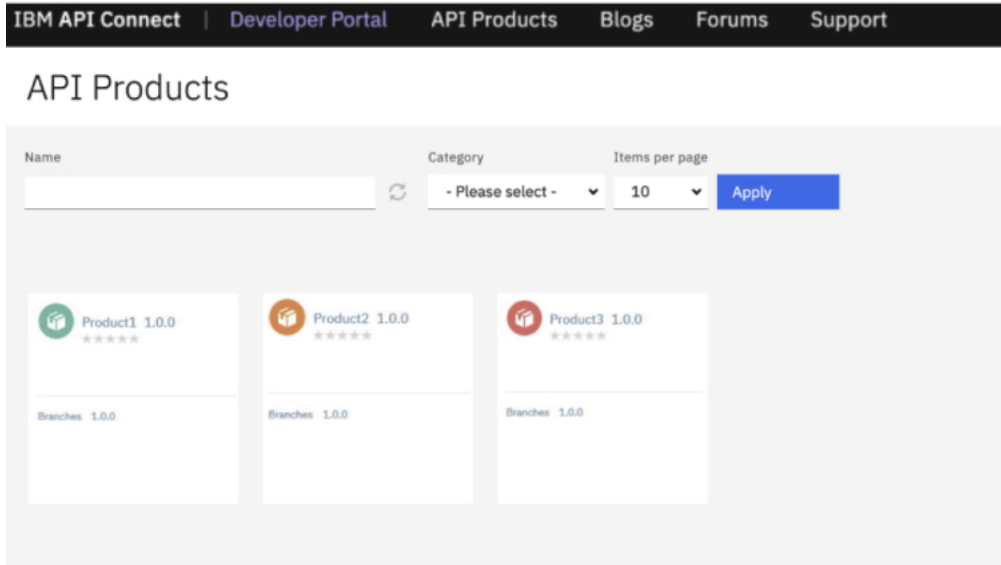
<input type="checkbox"/>	TITLE	CONTENT TYPE	AUTHOR	STATUS	UPDATED	OPERATIONS
<input type="checkbox"/>	Product3	Product	admin	Published	06/18/2019 - 15:34	Edit
<input type="checkbox"/>	Product2	Product	admin	Published	06/18/2019 - 15:32	Edit
<input type="checkbox"/>	Product1	Product	admin	Published	06/18/2019 - 15:31	Edit

3. For Product1 click edit, in the **Weighting** field, select 2 as the value, click Save.
4. For Product2 click edit, in the **Weighting** field, select 3 as the value, click Save.
5. For Product3 click edit, in the **Weighting** field, select 1 as the value, click Save.

The API Products page displays Product3, Product1, and Product2 in this order after the weightings are added.

## What you did in this tutorial

You changed the order that the API Products appear on the products page.



## Tutorial: Synchronizing application credentials from the Developer Portal with an external server

You can synchronize application credentials from the Developer Portal with an external server.

### Before you begin

You must have administrator access to complete this task.

Download the [Application credential synchronization](#) module.

### About this task

In this tutorial, you can synchronize the application credentials from the Developer Portal with an external credentials server that your organization might already have. This is done by configuring a custom module and installing it in the Developer Portal.

The file that you have downloaded is a custom module that contains multiple methods that can be configured with your REST endpoints, and any further specifications that you want to define, to synchronize your Developer Portal application credentials with an external credential server.

For more information, see [Custom module development: background and prerequisites](#).

### Procedure

To configure the custom module to synchronize application credentials:

1. In the `appcreds_sync.module` file, replace `example.com` with your REST endpoint for the following method types:
  - `http://example.com/app/create`
  - `http://example.com/app/delete`
  - `http://example.com/app/creds/create`
  - `http://example.com/app/creds/update`
  - `http://example.com/app/creds/delete`
  - `http://example.com/app/clientid/reset`
  - `http://example.com/app/clientsecret/reset`
2. Optional: Replace `username` and `password` with your username and password in the `_appcreds_sync_json_http_request` function:

```
('username' . ':' . 'password')
```

Note: You can also configure the SSL settings if you need to.

3. After you have finished configuring the module, click Save, and zip up the module.

To install the configured custom module in the Developer Portal:

4. On the administrator dashboard, click Modules
5. Click Install new modules.
6. You can enter a path to the module in the Install from a URL field. Alternatively, you can upload a module under the Upload a module or theme archive to install heading.
7. Click Install.

To enable the module:

8. On the administrator dashboard, click Modules.
9. Search for and enable the custom module.
10. Click Save configuration.

## Results

You have configured a custom module that synchronizes application credentials from the Developer Portal, installed that module in the Developer Portal, and enabled it. No further action or configuration is required.

## Reference information

Reference information for IBM® API Connect, including summary reference material for the developer toolkit command-line tool (CLI), a link to the API Connect Version 10.x Whitepaper, and information about the API Connect platform REST APIs.

Reference item	Description
<a href="#">Command-line tool reference for the developer toolkit</a>	Summary reference material for each of the available commands for the developer toolkit command-line tool (CLI). (For more detailed information about how to use the command-line tool, see <a href="#">Using the developer toolkit command-line tool</a> .)
<a href="#">Command-line tool reference for the Developer Portal</a>	Summary reference material for each of the available commands for the Developer Portal command-line tool (CLI). (For more detailed information about how to use the command-line tool, see <a href="#">Using the Developer Portal command-line tool</a> .)
<a href="#">API Connect REST APIs</a>	The platform REST APIs that can be used to manage the API Connect cloud configuration, access the API provider capability, and use the consumer API microservice.
<a href="#">API Connect v10.x Whitepaper</a>	A comprehensive technical guide to best practices, considerations, and deployment options for API Connect. The whitepaper covers the major components of API Connect, as well as considerations for configuring different clouds and environments, to help users with their API strategies. The target audience for the whitepaper are solution and integration architects.

## apic

APIConnect toolkit a7bbaf073f79ac4f140fb3d78c7a8937e3d1fde5 (Built 2023-06-02T18:13:39Z)

## Synopsis

APIConnect toolkit a7bbaf073f79ac4f140fb3d78c7a8937e3d1fde5 (Built 2023-06-02T18:13:39Z)

`apic [flags]`

## Options

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
-h, --help Help for apic
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic activations

Activations operations

## Synopsis

Activations operations

`apic activations [flags]`

## Options

```

-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for activations
--limit int32 Maximum number of items to return

```

```
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
```

---

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic activations:clear

Activations clear operations

---

## Synopsis

---

Activations clear operations

```
apic activations:clear [flags]
```

---

## Options

---

```
-c, --catalog string Catalog name or id (required)
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for activations:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
```

---

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic activations:delete

Activations delete operations

---

## Synopsis

---

Activations delete operations

```
apic activations:delete [flags]
```

---

## Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for activations:delete
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
```

---

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic activations:get

Activations get operations

## Synopsis

---

Activations get operations

```
apic activations:get [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for activations:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic activations:list

Activations list operations

## Synopsis

---

Activations list operations

```
apic activations:list [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for activations:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic analytics

Analytics operations

## Synopsis

---

Analytics operations

```
apic analytics [flags]
```

## Options

---

```
-h, --help Help for analytics
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
```



```
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic analytics:create

Analytics create operations

---

### Synopsis

Analytics create operations

```
apic analytics:create [flags]
```

---

### Options

```
--analytics-service string Analytics Service name or id (required)
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for analytics:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--query string Add query to request
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic analytics-services

Analytics Services operations

---

### Synopsis

Analytics Services operations

```
apic analytics-services [flags]
```

---

### Options

```
--availability-zone string Availability Zone name or id (required)
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for analytics-services
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic analytics-services:clear

Clear the Analytics Service objects

---

### Synopsis

Clear the Analytics Service objects

```
apic analytics-services:clear [flags]
```

## Options

---

<code>--availability-zone</code> string	Availability Zone name or id (required)
<code>--confirm</code> string	Confirmation for critical updates (required)
<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for analytics-services:clear
<code>-o, --org</code> string	Organization name or id (required)
<code>--output</code> string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code> string	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic analytics-services:create

Create a Analytics Service object

## Synopsis

---

Create a Analytics Service object Fields not allowed:\ - gateway\_service\_urls - ingestion\_endpoint - client\_endpoint Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - availability\_zone\_url

```
apic analytics-services:create [flags]
```

## Options

---

<code>--availability-zone</code> string	Availability Zone name or id (required)
<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for analytics-services:create
<code>-o, --org</code> string	Organization name or id (required)
<code>--output</code> string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code> string	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic analytics-services:delete

Delete the Analytics Service object by name or id

## Synopsis

---

Delete the Analytics Service object by name or id

```
apic analytics-services:delete [flags]
```

## Options

---

<code>--availability-zone</code> string	Availability Zone name or id (required)
<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for analytics-services:delete
<code>-o, --org</code> string	Organization name or id (required)
<code>--output</code> string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code> string	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

---

## apic analytics-services:get

Get the Analytics Service object by name or id

### Synopsis

---

Get the Analytics Service object by name or id

```
apic analytics-services:get [flags]
```

### Options

---

```
--availability-zone string Availability Zone name or id (required)
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for analytics-services:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--query string Add query to request
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic analytics-services:list

List the Analytics Service objects

### Synopsis

---

List the Analytics Service objects

```
apic analytics-services:list [flags]
```

### Options

---

```
--availability-zone string Availability Zone name or id (required)
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for analytics-services:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic analytics-services:update

Update the Analytics Service object by name or id

### Synopsis

---

Update the Analytics Service object by name or id Fields not allowed to be null:\ - ingestion\_endpoint\_tls\_client\_profile\_url - client\_endpoint\_tls\_client\_profile\_url Fields not allowed:\ - endpoint - ingestion\_endpoint - client\_endpoint Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - availability\_zone\_url

```
apic analytics-services:update [flags]
```

## Options

---

```
--availability-zone string Availability Zone name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for analytics-services:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic api-keys

Api Keys operations

## Synopsis

---

Api Keys operations

```
apic api-keys [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for api-keys
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic api-keys:create

Create a API Key object

## Synopsis

---

Create a API Key object Required fields:\ - client\_type - description Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url

```
apic api-keys:create [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for api-keys:create
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic api-keys:delete

Delete the API Key object by name or id

## Synopsis

---

Delete the API Key object by name or id

```
apic api-keys:delete [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for api-keys:delete
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic api-keys:get

Get the API Key object by name or id

## Synopsis

---

Get the API Key object by name or id

```
apic api-keys:get [flags]
```

## Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for api-keys:get
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic api-keys:list

List the API Key objects

## Synopsis

---

List the API Key objects

```
apic api-keys:list [flags]
```

## Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for api-keys:list
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file

```
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic apis

Apis operations

---

### Synopsis

Apis operations

```
apic apis [flags]
```

---

### Options

```
--api_type string The type of api (asynccapi, rest, graphql, wsd1_to_rest, or wsd1)
--base_path string Base path of the API
-c, --catalog string Catalog name or id (required)
--enforced Whether the API is enforced or not
--expand string List of transient field to expand
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for apis
--limit int32 Maximum number of items to return
--metadata string List of metadata fields in the api to filter on
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic apis:clone

Apis clone operations

---

### Synopsis

Apis clone operations

```
apic apis:clone [flags]
```

---

### Options

```
--api_type string The type of api (asynccapi, rest, graphql, wsd1_to_rest, or wsd1)
--base_path string Base path of the API
-c, --catalog string Catalog name or id (required)
--enforced Whether the API is enforced or not
--expand string List of transient field to expand
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for apis:clone
--limit int32 Maximum number of items to return
--metadata string List of metadata fields in the api to filter on
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic apis:document

Apis document operations

### Synopsis

---

Apis document operations

```
apic apis:document [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for apis:document
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic apis:get

Apis get operations

### Synopsis

---

Apis get operations

```
apic apis:get [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return (default "add(wsd,api)")
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for apis:get
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic apis:list

Apis list operations

### Synopsis

---

Apis list operations

```
apic apis:list [flags]
```

### Options

---

```

--api_type string The type of api (asynccapi, rest, graphql, wsd1_to_rest, or wsd1)
--base_path string Base path of the API
-c, --catalog string Catalog name or id (required)
--enforced Whether the API is enforced or not
--expand string List of transient field to expand
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for apis:list
--limit int32 Maximum number of items to return
--metadata string List of metadata fields in the api to filter on
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic apis:list-all

Apis list-all operations

### Synopsis

Apis list-all operations

```
apic apis:list-all [flags]
```

### Options

```

--api_type string The type of api (asynccapi, rest, graphql, wsd1_to_rest, or wsd1)
--base_path string Base path of the API
-c, --catalog string Catalog name or id (required)
--enforced Whether the API is enforced or not
--expand string List of transient field to expand
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for apis:list-all
--limit int32 Maximum number of items to return
--metadata string List of metadata fields in the api to filter on
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic apis:update

Apis update operations

### Synopsis

Apis update operations

```
apic apis:update [flags]
```

### Options

```

-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for apis:update
--id string id
-o, --org string Organization name or id (required)

```



```
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic apis:validate

Validate the draft api object

## Synopsis

---

Validate the draft api object

```
apic apis:validate [flags]
```

## Options

---

```
--catalog_name string Name of a catalog
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
--gateway_service_names string Names of Gateway Services
-h, --help Help for apis:validate
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space_name string Name of a space
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic apis:wSDL

Apis wsdL operations

## Synopsis

---

Apis wsdL operations

```
apic apis:wSDL [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for apis:wSDL
--id string id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic apps

## Synopsis

---

Apps operations

```
apic apps [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--consumer-org string Consumer Organization name or id (required)
--expand string List of transient field to expand
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for apps
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic apps:clear

Clear the Application objects

## Synopsis

---

Clear the Application objects

```
apic apps:clear [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--confirm string Confirmation for critical updates (required)
--consumer-org string Consumer Organization name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for apps:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic apps:create

Create a Application object

## Synopsis

---

Create a Application object Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - catalog\_url - consumer\_org\_url

```
apic apps:create [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for apps:create
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic apps:delete

Delete the Application object by name or id

### Synopsis

---

Delete the Application object by name or id

```
apic apps:delete [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for apps:delete
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic apps:get

Get the Application object by name or id

### Synopsis

---

Get the Application object by name or id

```
apic apps:get [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for apps:get
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic apps:list

List the Application objects

### Synopsis

---

List the Application objects

```
apic apps:list [flags]
```

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--consumer-org string</code>	Consumer Organization name or id (required)
<code>--expand string</code>	List of transient field to expand
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for apps:list
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic apps:update

Update the Application object by name or id

### Synopsis

---

Update the Application object by name or id Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - catalog\_url - consumer\_org\_url

```
apic apps:update [flags]
```

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--consumer-org string</code>	Consumer Organization name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for apps:update
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic associates

Associates operations

### Synopsis

---

Associates operations

apic associates [flags]

## Options

---

-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for associates
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic associates:get

Associates get operations

## Synopsis

---

Associates get operations

apic associates:get [flags]

## Options

---

-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for associates:get
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic associates:list

Associates list operations

## Synopsis

---

Associates list operations

apic associates:list [flags]

## Options

---

-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for associates:list
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)

<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic availability-zones

Availability Zones operations

### Synopsis

---

Availability Zones operations

`apic availability-zones [flags]`

### Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for availability-zones
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic availability-zones:clear

Clear the Availability Zone objects

### Synopsis

---

Clear the Availability Zone objects

`apic availability-zones:clear [flags]`

### Options

---

<code>--confirm string</code>	Confirmation for critical updates (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for availability-zones:clear
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic availability-zones:create

Create a Availability Zone object

## Synopsis

---

Create a Availability Zone object Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url

```
apic availability-zones:create [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for availability-zones:create
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic availability-zones:delete

Delete the Availability Zone object by name or id

## Synopsis

---

Delete the Availability Zone object by name or id

```
apic availability-zones:delete [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for availability-zones:delete
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic availability-zones:get

Get the Availability Zone object by name or id

## Synopsis

---

Get the Availability Zone object by name or id

```
apic availability-zones:get [flags]
```

## Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for availability-zones:get
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic availability-zones:list

List the Availability Zone objects

### Synopsis

---

List the Availability Zone objects

```
apic availability-zones:list [flags]
```

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for availability-zones:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic availability-zones:update

Update the Availability Zone object by name or id

### Synopsis

---

Update the Availability Zone object by name or id Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url

```
apic availability-zones:update [flags]
```

### Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for availability-zones:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic billings

Billings operations

### Synopsis

---

Billings operations

```
apic billings [flags]
```

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for billings
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
```



```
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic billings:clear

Clear the Billing objects

### Synopsis

---

Clear the Billing objects

```
apic billings:clear [flags]
```

### Options

---

```
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for billings:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic billings:delete

Delete the Billing object by name or id

### Synopsis

---

Delete the Billing object by name or id

```
apic billings:delete [flags]
```

### Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for billings:delete
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic billings:get

Get the Billing object by name or id

### Synopsis

---

Get the Billing object by name or id

```
apic billings:get [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for billings:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic billings:list

List the Billing objects

## Synopsis

---

List the Billing objects

```
apic billings:list [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for billings:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic billings:update

Update the Billing object by name or id

## Synopsis

---

Update the Billing object by name or id Fields not allowed to be null:\ - integration\_url Fields not allowed:\ - job\_queue\_status Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url

```
apic billings:update [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for billings:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic catalog-settings

Catalog Settings operations

### Synopsis

---

Catalog Settings operations

```
apic catalog-settings [flags]
```

### Options

---

```
-h, --help Help for catalog-settings
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic catalog-settings:get

Get the Catalog Setting object

### Synopsis

---

Get the Catalog Setting object

```
apic catalog-settings:get [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for catalog-settings:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic catalog-settings:update

Update the Catalog Setting object

### Synopsis

---

Update the Catalog Setting object Fields allowed but ignored:\ - name - type - api\_version - scope - created\_at - updated\_at - url - org\_url - catalog\_url

```
apic catalog-settings:update [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--force Force the operation
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for catalog-settings:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic catalogs

Catalogs operations

---

### Synopsis

Catalogs operations

```
apic catalogs [flags]
```

---

### Options

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for catalogs
--limit int32 Maximum number of items to return
--my my
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic catalogs:clear

Clear the Catalog objects

---

### Synopsis

Clear the Catalog objects

```
apic catalogs:clear [flags]
```

---

### Options

```
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for catalogs:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic catalogs:create

Create a Catalog object

---

### Synopsis

Create a Catalog object Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url

```
apic catalogs:create [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for catalogs:create
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic catalogs:delete

Delete the Catalog object by name or id

## Synopsis

---

Delete the Catalog object by name or id

```
apic catalogs:delete [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for catalogs:delete
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic catalogs:email-to-owners

Send email to owners of consumer organizations

## Synopsis

---

Send email to owners of consumer organizations, given consumer org and consumer group urls. For consumer group, email owners of all consumer orgs in the group.

```
apic catalogs:email-to-owners [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for catalogs:email-to-owners
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic catalogs:get

Get the Catalog object by name or id

## Synopsis

---

Get the Catalog object by name or id

```
apic catalogs:get [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for catalogs:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic catalogs:list

List the Catalog objects

## Synopsis

---

List the Catalog objects

```
apic catalogs:list [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for catalogs:list
--limit int32 Maximum number of items to return
--my my
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic catalogs:transfer-owner

Transfer owner to an associate

## Synopsis

---

Transfer owner to an associate

```
apic catalogs:transfer-owner [flags]
```

## Options

---

```
--cascade Cascade the behavior
--delete_old_owner Delete old owner
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for catalogs:transfer-owner
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic catalogs:update

Update the Catalog object by name or id

### Synopsis

Update the Catalog object by name or id Fields not allowed:\ - name Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url

```
apic catalogs:update [flags]
```

### Options

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for catalogs:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic client-creds

Manage the client configuration parameters. Client ID and Client Secret.

### Synopsis

Manage the client configuration parameters. Client ID and Client Secret.

```
apic client-creds [flags]
```

### Options

```
-g, --global list the global configuration variables
-h, --help Help for client-creds
-l, --local list the local application configuration variables
```

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic client-creds:clear

clear the set client credentials.

### Synopsis

clear the set client credentials.

```
apic client-creds:clear [flags]
```

### Examples

```
$ apic client-creds:clear
Deleted client credentials.
```

## Options

---

```
-g, --global list the global configuration variables
-h, --help Help for client-creds:clear
-l, --local list the local application configuration variables
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic client-creds:list

List the set clientID/secret parameters.

## Synopsis

---

List the set clientID/secret parameters.

```
apic client-creds:list [flags]
```

## Examples

---

```
$ apic client-creds:list
```

## Options

---

```
-g, --global list the global configuration variables
-h, --help Help for client-creds:list
-l, --local list the local application configuration variables
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic client-creds:set

Set the client configuration parameters. ClientID and Client Secret.

## Synopsis

---

Set the client configuration parameters. ClientID and Client Secret.

```
apic client-creds:set /path/to/creds/json ... [flags]
```

## Examples

---

```
$ apic client-creds:set /Users/local_user/credential.json
```

## Options

---

```
-g, --global list the global configuration variables
-h, --help Help for client-creds:set
-l, --local list the local application configuration variables
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic cloud-emails



## Synopsis

---

Cloud Emails operations

```
apic cloud-emails [flags]
```

## Options

---

```
-h, --help Help for cloud-emails
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic cloud-emails:send-to-owners

---

Email provider org owners

## Synopsis

---

Send an email to owners of provider orgs

```
apic cloud-emails:send-to-owners [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for cloud-emails:send-to-owners
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic cloud-settings

---

Cloud Settings operations

## Synopsis

---

Cloud Settings operations

```
apic cloud-settings [flags]
```

## Options

---

```
-h, --help Help for cloud-settings
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic cloud-settings:about

---

Return information about the cloud

## Synopsis

---

Return public information about the cloud. You MUST supply the --format option when using this command from the API Connect toolkit.

```
apic cloud-settings:about [flags]
```

## Options

---

--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for cloud-settings:about
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic cloud-settings:audit-endpoint-test-connection

---

Test an audit endpoint connection

## Synopsis

---

Test an audit endpoint connection

```
apic cloud-settings:audit-endpoint-test-connection [flags]
```

## Options

---

--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for cloud-settings:audit-endpoint-test-connection
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic cloud-settings:designer-credentials-list

---

List credential for designer

## Synopsis

---

List credential for designer

```
apic cloud-settings:designer-credentials-list [flags]
```

## Options

---

--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for cloud-settings:designer-credentials-list
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic cloud-settings:get

---

Get the Cloud Setting object

## Synopsis

---

Get the Cloud Setting object

```
apic cloud-settings:get [flags]
```

## Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for cloud-settings:get
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic cloud-settings:info

---

Return public information about the cloud deployment

## Synopsis

---

Return public information about the cloud deployment

```
apic cloud-settings:info [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for cloud-settings:info
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic cloud-settings:mail-server-configured

---

Return true or false based on if mail server is configured or not

## Synopsis

---

Return true or false based on if mail server is configured or not. You MUST supply the --format option when using this command from the API Connect toolkit.

```
apic cloud-settings:mail-server-configured [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for cloud-settings:mail-server-configured
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic cloud-settings:oauth2-certs

Support JWKS\_URI endpoint, conform to OIDC specification

### Synopsis

---

Return JWKS that is used to issue token. You MUST supply the --format option when using this command from the API Connect toolkit.

```
apic cloud-settings:oauth2-certs [flags]
```

### Options

---

--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for cloud-settings:oauth2-certs
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)

### Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic cloud-settings:toolkit-credentials-list

List credential for toolkit and consumer toolkit

### Synopsis

---

List credential for toolkit and consumer toolkit

```
apic cloud-settings:toolkit-credentials-list [flags]
```

### Options

---

--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for cloud-settings:toolkit-credentials-list
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)

### Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic cloud-settings:topology

Return the topology of the cloud for governance

### Synopsis

---

Return the topology of the cloud for governance. You MUST supply the --format option when using this command from the API Connect toolkit.

```
apic cloud-settings:topology [flags]
```

### Options

---

--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for cloud-settings:topology
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)

### Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output

```
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic cloud-settings:update

Update the Cloud Setting object

### Synopsis

Update the Cloud Setting object Fields not allowed to be null:\ - email\_sender - tls\_client\_profile\_default\_url Fields not allowed:\ - service\_type - cloud\_id - ibm\_cloud  
Fields allowed but ignored:\ - name - type - api\_version - scope - created\_at - updated\_at - url

```
apic cloud-settings:update [flags]
```

### Options

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for cloud-settings:update
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic config

Manage configuration variables

### Synopsis

Manage configuration variables

```
apic config [flags]
```

### Options

```
-g, --global list the global configuration variables
-h, --help Help for config
-l, --local list the local application configuration variables
```

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic config:clear

Delete all configuration variables

### Synopsis

Delete all configuration variables

```
apic config:clear [flags]
```

### Examples

```
$ apic config:clear
Deleted config catalog
Deleted config org
Deleted config space
```

```
$ apic config:clear --global
Deleted config catalog
```

## Options

---

```
-g, --global list the global configuration variables
-h, --help Help for config:clear
-l, --local list the local application configuration variables
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic config:delete

Delete a configuration variable

## Synopsis

---

Configuration variables:

catalog The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the --server, --organization, and --catalog command line options. The catalog URI has the form: `https://MANAGEMENT_SERVER/api/catalogs/ORGANIZATION_NAME/CATALOG_NAME`

org The default value of org defined by the app's or catalog's URI can be set using the org URI. The org URI has the form: `https://MANAGEMENT_SERVER/api/orgs/ORGANIZATION_NAME`

space The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the --server, --organization, --catalog, and --space command line options. The space URI has the form: `https://MANAGEMENT_SERVER/api/spaces/ORGANIZATION_NAME/CATALOG_NAME/SPACE_NAME`

consumer The consumer configuration variable should be set to the URI of an API Connect consumer. The value of the consumer-org provides the default identity of a consumer. The default values defined by the consumer URI can be overridden using the --server, --organization, --catalog, and --consumer command line options. The consumer URI has the form: `https://MANAGEMENT_SERVER/api/consumer-orgs/ORGANIZATION_NAME/CATALOG_NAME/CONSUMER_ORG_NAME`

cloud The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the --server and --mode command line options. The cloud URI has the form: `https://MANAGEMENT_SERVER/api/`

mode The value of mode configuration variable defines the toolkit operation mode. It can be overridden using the --mode command line option. The value can be set to apim or consumer

template-path List of places to look for handlebar templates

template-default-api Defines the default api template

template-default-product Defines the default product template

```
apic config:delete NAME... [flags]
```

## Examples

---

```
$ apic config:delete catalog
Deleted config catalog
```

```
$ apic config:delete org
Deleted config org
```

```
$ apic config:delete template-path
Deleted config template-path
```

## Options

---

```
-g, --global list the global configuration variables
-h, --help Help for config:delete
-l, --local list the local application configuration variables
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic config:get

Get a configuration variable

### Synopsis

---

Configuration variables:

**catalog** The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the `--server`, `--organization`, and `--catalog` command line options. The catalog URI has the form: `https://MANAGEMENT_SERVER/api/catalogs/ORGANIZATION_NAME/CATALOG_NAME`

**org** The default value of `org` defined by the app's or catalog's URI can be set using the `org` URI. The `org` URI has the form: `https://MANAGEMENT_SERVER/api/orgs/ORGANIZATION_NAME`

**space** The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the `--server`, `--organization`, `--catalog`, and `--space` command line options. The space URI has the form: `https://MANAGEMENT_SERVER/api/spaces/ORGANIZATION_NAME/CATALOG_NAME/SPACE_NAME`

**consumer** The consumer configuration variable should be set to the URI of an API Connect consumer. The value of the consumer-`org` provides the default identity of a consumer. The default values defined by the consumer URI can be overridden using the `--server`, `--organization`, `--catalog`, and `--consumer` command line options. The consumer URI has the form: `https://MANAGEMENT_SERVER/api/consumer-orgs/ORGANIZATION_NAME/CATALOG_NAME/CONSUMER_ORG_NAME`

**cloud** The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the `--server` and `--mode` command line options. The cloud URI has the form: `https://MANAGEMENT_SERVER/api/`

**mode** The value of `mode` configuration variable defines the toolkit operation mode. It can be overridden using the `--mode` command line option. The value can be set to `apim` or `consumer`

**template-path** List of places to look for handlebar templates

**template-default-api** Defines the default api template

**template-default-product** Defines the default product template

```
apic config:get NAME [flags]
```

### Examples

---

```
$ apic config:get catalog
catalog: https://mgmthost.com/api/catalogs/climbon/sb
```

```
$ apic config:get org
org: https://mgmthost.com/api/orgs/climbon
```

```
$ apic config:get template-path
template-path: /etc/templates
```

### Options

---

```
-g, --global list the global configuration variables
-h, --help Help for config:get
-l, --local list the local application configuration variables
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic config:list

List the application and global configuration variables

### Synopsis

---

Configuration variables:

**catalog** The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the `--server`, `--organization`, and `--catalog` command line options. The catalog URI has the form: `https://MANAGEMENT_SERVER/api/catalogs/ORGANIZATION_NAME/CATALOG_NAME`

**org** The default value of `org` defined by the app's or catalog's URI can be set using the `org` URI. The `org` URI has the form: `https://MANAGEMENT_SERVER/api/orgs/ORGANIZATION_NAME`

space The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the --server, --organization, --catalog, and --space command line options. The space URI has the form: https://MANAGEMENT\_SERVER/api/spaces/ORGANIZATION\_NAME/CATALOG\_NAME/SPACE\_NAME

consumer The consumer configuration variable should be set to the URI of an API Connect consumer. The value of the consumer-org provides the default identity of a consumer. The default values defined by the consumer URI can be overridden using the --server, --organization, --catalog, and --consumer command line options. The consumer URI has the form: https://MANAGEMENT\_SERVER/api/consumer-orgs/ORGANIZATION\_NAME/CATALOG\_NAME/CONSUMER\_ORG\_NAME

cloud The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the --server and --mode command line options. The cloud URI has the form: https://MANAGEMENT\_SERVER/api/

mode The value of mode configuration variable defines the toolkit operation mode. It can be overridden using the --mode command line option. The value can be set to apim or consumer

template-path List of places to look for handlebar templates

template-default-api Defines the default api template

template-default-product Defines the default product template

**apic config:list [flags]**

## Examples

```
$ apic config
catalog: https://mgmthost.com/api/catalogs/climbon/sb
org: https://mgmthost.com/api/orgs/climbon
```

```
$ apic config --global
catalog: https://mgmthost.com/api/catlogs/climbon/sb
```

## Options

```
-g, --global list the global configuration variables
-h, --help Help for config:list
-l, --local list the local application configuration variables
```

## Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic config:set

Set or update configuration variables

## Synopsis

Configuration variables:

catalog The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the --server, --organization, and --catalog command line options. The catalog URI has the form: https://MANAGEMENT\_SERVER/api/catalogs/ORGANIZATION\_NAME/CATALOG\_NAME

org The default value of org defined by the app's or catalog's URI can be set using the org URI. The org URI has the form: https://MANAGEMENT\_SERVER/api/orgs/ORGANIZATION\_NAME

space The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the --server, --organization, --catalog, and --space command line options. The space URI has the form: https://MANAGEMENT\_SERVER/api/spaces/ORGANIZATION\_NAME/CATALOG\_NAME/SPACE\_NAME

consumer The consumer configuration variable should be set to the URI of an API Connect consumer. The value of the consumer-org provides the default identity of a consumer. The default values defined by the consumer URI can be overridden using the --server, --organization, --catalog, and --consumer command line options. The consumer URI has the form: https://MANAGEMENT\_SERVER/api/consumer-orgs/ORGANIZATION\_NAME/CATALOG\_NAME/CONSUMER\_ORG\_NAME

cloud The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the --server and --mode command line options. The cloud URI has the form: https://MANAGEMENT\_SERVER/api/

mode The value of mode configuration variable defines the toolkit operation mode. It can be overridden using the --mode command line option. The value can be set to apim or consumer

template-path List of places to look for handlebar templates

template-default-api Defines the default api template

template-default-product Defines the default product template

**apic config:set NAME=VALUE ... [flags]**



## Examples

---

```
$ apic config:set catalog=https://mgmthost.com/api/catalogs/climbon/sb
catalog: https://mgmthost.com/api/catalogs/climbon/sb

$ apic config:set org=https://mgmthost2.com/api/orgs/hikeon
org: https://mgmthost2.com/api/orgs/hikeon

$ apic config:set --global template-path="/etc/templates"
template-path: /etc/templates
```

## Options

---

```
-g, --global list the global configuration variables
-h, --help Help for config:set
-l, --local list the local application configuration variables
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic configured-api-user-registries

---

Configured Api User Registries operations

## Synopsis

---

Configured Api User Registries operations

```
apic configured-api-user-registries [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-api-user-registries
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic configured-api-user-registries:clear

---

Configured Api User Registries clear operations

## Synopsis

---

Configured Api User Registries clear operations

```
apic configured-api-user-registries:clear [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-api-user-registries:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
```

```
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic configured-api-user-registries:create

---

Configured Api User Registries create operations

### Synopsis

---

Configured Api User Registries create operations

```
apic configured-api-user-registries:create [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-api-user-registries:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic configured-api-user-registries:delete

---

Configured Api User Registries delete operations

### Synopsis

---

Configured Api User Registries delete operations

```
apic configured-api-user-registries:delete [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-api-user-registries:delete
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic configured-api-user-registries:get

---

Configured Api User Registries get operations

## Synopsis

---

Configured Api User Registries get operations

```
apic configured-api-user-registries:get [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for configured-api-user-registries:get
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic configured-api-user-registries:list

---

Configured Api User Registries list operations

## Synopsis

---

Configured Api User Registries list operations

```
apic configured-api-user-registries:list [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for configured-api-user-registries:list
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic configured-billings

---

Configured Billings operations

## Synopsis

---

Configured Billings operations

```
apic configured-billings [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for configured-billings
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return

```
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic configured-billings:clear

Clear the Configured Billing objects

### Synopsis

---

Clear the Configured Billing objects

```
apic configured-billings:clear [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-billings:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic configured-billings:create

Create a Configured Billing object

### Synopsis

---

Create a Configured Billing object Required fields:\ - billing\_url Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - catalog\_url

```
apic configured-billings:create [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-billings:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic configured-billings:delete

Delete the Configured Billing object by name or id

## Synopsis

---

Delete the Configured Billing object by name or id

```
apic configured-billings:delete [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-billings:delete
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic configured-billings:get

Get the Configured Billing object by name or id

## Synopsis

---

Get the Configured Billing object by name or id

```
apic configured-billings:get [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-billings:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic configured-billings:list

List the Configured Billing objects

## Synopsis

---

List the Configured Billing objects

```
apic configured-billings:list [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-billings:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic configured-catalog-user-registries

Configured Catalog User Registries operations

---

### Synopsis

Configured Catalog User Registries operations

```
apic configured-catalog-user-registries [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-catalog-user-registries
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic configured-catalog-user-registries:create

Create a Configured Catalog User Registry object

---

### Synopsis

Create a Configured Catalog User Registry object Required fields:\ - user\_registry\_url Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - catalog\_url - user\_registry\_url

```
apic configured-catalog-user-registries:create [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-catalog-user-registries:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic configured-catalog-user-registries:delete

Delete the Configured Catalog User Registry object by name or id

---

### Synopsis

Delete the Configured Catalog User Registry object by name or id

```
apic configured-catalog-user-registries:delete [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for configured-catalog-user-registries:delete
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic configured-catalog-user-registries:get

Get the Configured Catalog User Registry object by name or id

## Synopsis

---

Get the Configured Catalog User Registry object by name or id

```
apic configured-catalog-user-registries:get [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for configured-catalog-user-registries:get
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic configured-catalog-user-registries:list

List the Configured Catalog User Registry objects

## Synopsis

---

List the Configured Catalog User Registry objects

```
apic configured-catalog-user-registries:list [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for configured-catalog-user-registries:list
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic configured-catalog-user-registries:search

Search for users in the catalog user registry

### Synopsis

---

Search for users in the catalog user registry

```
apic configured-catalog-user-registries:search [flags]
```

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for configured-catalog-user-registries:search
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic configured-gateway-services

Configured Gateway Services operations

### Synopsis

---

Configured Gateway Services operations

```
apic configured-gateway-services [flags]
```

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for configured-gateway-services
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic configured-gateway-services:clear

Configured Gateway Services clear operations

### Synopsis

---

Configured Gateway Services clear operations

```
apic configured-gateway-services:clear [flags]
```

### Options

---



-c, --catalog string	Catalog name or id (required)
--confirm string	Confirmation for critical updates (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for configured-gateway-services:clear
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic configured-gateway-services:create

---

Configured Gateway Services create operations

### Synopsis

---

Configured Gateway Services create operations

```
apic configured-gateway-services:create [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for configured-gateway-services:create
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic configured-gateway-services:delete

---

Configured Gateway Services delete operations

### Synopsis

---

Configured Gateway Services delete operations

```
apic configured-gateway-services:delete [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for configured-gateway-services:delete
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic configured-gateway-services:get

Configured Gateway Services get operations

### Synopsis

---

Configured Gateway Services get operations

```
apic configured-gateway-services:get [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-gateway-services:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic configured-gateway-services:list

Configured Gateway Services list operations

### Synopsis

---

Configured Gateway Services list operations

```
apic configured-gateway-services:list [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-gateway-services:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic configured-oauth-providers

Configured OAuth Providers operations

### Synopsis

---

Configured OAuth Providers operations

```
apic configured-oauth-providers [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for configured-oauth-providers
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic configured-oauth-providers:clear

Configured OAuth Providers clear operations

### Synopsis

---

Configured OAuth Providers clear operations

```
apic configured-oauth-providers:clear [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--confirm string	Confirmation for critical updates (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for configured-oauth-providers:clear
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic configured-oauth-providers:create

Configured OAuth Providers create operations

### Synopsis

---

Configured OAuth Providers create operations

```
apic configured-oauth-providers:create [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for configured-oauth-providers:create
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic configured-oauth-providers:delete

Configured OAuth Providers delete operations

### Synopsis

---

Configured OAuth Providers delete operations

```
apic configured-oauth-providers:delete [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-oauth-providers:delete
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic configured-oauth-providers:get

Configured OAuth Providers get operations

### Synopsis

---

Configured OAuth Providers get operations

```
apic configured-oauth-providers:get [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-oauth-providers:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic configured-oauth-providers:list

Configured OAuth Providers list operations

### Synopsis

---

Configured OAuth Providers list operations

```
apic configured-oauth-providers:list [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
```

```

--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-oauth-providers:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

---

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic configured-tls-client-profiles

Configured Tls Client Profiles operations

### Synopsis

Configured Tls Client Profiles operations

```
apic configured-tls-client-profiles [flags]
```

### Options

```

-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-tls-client-profiles
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

---

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic configured-tls-client-profiles:clear

Configured Tls Client Profiles clear operations

### Synopsis

Configured Tls Client Profiles clear operations

```
apic configured-tls-client-profiles:clear [flags]
```

### Options

```

-c, --catalog string Catalog name or id (required)
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-tls-client-profiles:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

---

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file

```

```
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic configured-tls-client-profiles:clear-all

Configured Tls Client Profiles clear-all operations

---

### Synopsis

Configured Tls Client Profiles clear-all operations

```
apic configured-tls-client-profiles:clear-all [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-tls-client-profiles:clear-all
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic configured-tls-client-profiles:create

Configured Tls Client Profiles create operations

---

### Synopsis

Configured Tls Client Profiles create operations

```
apic configured-tls-client-profiles:create [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-tls-client-profiles:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic configured-tls-client-profiles:delete

Configured Tls Client Profiles delete operations

---

### Synopsis

Configured Tls Client Profiles delete operations

```
apic configured-tls-client-profiles:delete [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for configured-tls-client-profiles:delete
--id	id
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic configured-tls-client-profiles:get

---

Configured Tls Client Profiles get operations

### Synopsis

---

Configured Tls Client Profiles get operations

```
apic configured-tls-client-profiles:get [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for configured-tls-client-profiles:get
--id	id
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic configured-tls-client-profiles:list

---

Configured Tls Client Profiles list operations

### Synopsis

---

Configured Tls Client Profiles list operations

```
apic configured-tls-client-profiles:list [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for configured-tls-client-profiles:list
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic configured-tls-client-profiles:list-all

Configured Tls Client Profiles list-all operations

---

### Synopsis

Configured Tls Client Profiles list-all operations

```
apic configured-tls-client-profiles:list-all [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for configured-tls-client-profiles:list-all
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic consumer-org-settings

Consumer Org Settings operations

---

### Synopsis

Consumer Org Settings operations

```
apic consumer-org-settings [flags]
```

---

### Options

```
-h, --help Help for consumer-org-settings
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic consumer-org-settings:delete

Delete the Consumer Organization Setting object

---

### Synopsis

Delete the Consumer Organization Setting object

```
apic consumer-org-settings:delete [flags]
```

---

### Options



-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for consumer-org-settings:delete
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic consumer-org-settings:get

Get the Consumer Organization Setting object

### Synopsis

Get the Consumer Organization Setting object

```
apic consumer-org-settings:get [flags]
```

### Options

-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for consumer-org-settings:get
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic consumer-org-settings:update

Update the Consumer Organization Setting object

### Synopsis

Update the Consumer Organization Setting object Fields allowed but ignored:\ - name - type - api\_version - scope - created\_at - updated\_at - url

```
apic consumer-org-settings:update [flags]
```

### Options

-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for consumer-org-settings:update
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic consumer-orgs

Consumer Orgs operations

### Synopsis

---

Consumer Orgs operations

```
apic consumer-orgs [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--expand string List of transient field to expand
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for consumer-orgs
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic consumer-orgs:clear

Clear the Consumer Organization objects

### Synopsis

---

Clear the Consumer Organization objects

```
apic consumer-orgs:clear [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for consumer-orgs:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic consumer-orgs:create

Create a Consumer Organization object

### Synopsis

---

Create a Consumer Organization object Required fields:\ - owner\_url Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - catalog\_url

```
apic consumer-orgs:create [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for consumer-orgs:create
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic consumer-orgs:delete

Delete the Consumer Organization object by name or id

## Synopsis

---

Delete the Consumer Organization object by name or id

```
apic consumer-orgs:delete [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for consumer-orgs:delete
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic consumer-orgs:get

Get the Consumer Organization object by name or id

## Synopsis

---

Get the Consumer Organization object by name or id

```
apic consumer-orgs:get [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for consumer-orgs:get
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file

```
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic consumer-orgs:list

List the Consumer Organization objects

---

### Synopsis

List the Consumer Organization objects

```
apic consumer-orgs:list [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--expand string List of transient field to expand
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for consumer-orgs:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic consumer-orgs:transfer-owner

Transfer owner to an associate

---

### Synopsis

Transfer owner to an associate

```
apic consumer-orgs:transfer-owner [flags]
```

---

### Options

```
--cascade Cascade the behavior
-c, --catalog string Catalog name or id (required)
--delete_old_owner Delete old owner
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for consumer-orgs:transfer-owner
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic consumer-orgs:update

Update the Consumer Organization object by name or id

---

### Synopsis

Update the Consumer Organization object by name or id Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - catalog\_url

```
apic consumer-orgs:update [flags]
```

## Options

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for consumer-orgs:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

## Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic create

Create an API or product definition

## Synopsis

Create an API or product definition

```
apic create [flags]
```

## Options

```
-h, --help Help for create
-i, --interactive use interactive mode
```

## Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic create:api

Create an OpenAPI (Swagger) definition

## Synopsis

Create an OpenAPI (Swagger) definition

```
apic create:api [flags]
```

## Examples

```
Create an API
$ apic create:api --title Routes
Created routes.yaml API definition [routes:1.0.0]

Create an API and generate a product referencing the API
$ apic create:api --title Routes --product "Climb On"
Created routes.yaml API definition [routes:1.0.0]
Created climb-on.yaml product definition [climb-on:1.0.0]

Create an API interactively
$ apic create:api
Title: Routes
Name (routes): routes
File (routes.yaml): routes.yaml
Template: ()
Basepath (/routes): /routes
Hostname ($(catalog.host)): $(catalog.host)
Schemes:
```

```

Target url: ()
Create product [true]: true
Product title (Routes Product): Climb On
Product name (climb-on): climb-on
Product file (climb-on.yaml): climb-on.yaml
Created routes.yaml API definition [routes:1.0.0]
Created climb-on.yaml product definition [climb-on:1.0.0]

Create an API from a WSDL document
$ apic create:api --wsdl globalweather.wsdl
Created globalweather.yaml API definition [globalweather.yaml:1.0.0]

Create an API using APIC's default OAuth 2 provider template
$ apic create:api --title "OAuth2 Provider" --template oauth2
Created oauth2-provider.yaml API definition [oauth2-provider:1.0.0]

Create an API using one of your templates
$ apic config:set --global template-path="/etc/templates"
$ ls /etc/templates
proxy.hbs staging.hbs
$ apic create:api --title "Proxy Provider" --template proxy
Created proxy-provider.yaml API definition [proxy-provider:1.0.0]

Create an API using your default template
$ apic config:set --global template-path="/etc/templates"
$ ls /etc/templates
proxy.hbs staging.hbs
$ apic config:set --global template-default-api=staging
$ apic create:api --title "Staging Provider"
Created staging-provider.yaml API definition [staging-provider:1.0.0]

```

## Options

```

--api_type string The type of api (rest, wsdl-to-rest, or wsdl) (default "wsdl")
--basepath string basepath value (default derived from name)
--disable_ws_security Disable generation of WS-Security definitions in api
--filename string filename (default derived from name)
--gateway-type string The type of the gateway (datapower-gateway, datapower-api-gateway, event-gateway) (default
"datapower-gateway")
-h, --help Help for create:api
--hostname string host value (default $(catalog.host))
-i, --interactive use interactive mode
--name string x-ibm-name value (default derived from title)
--product string generate a product definition referencing the API
--schemes string list of schemes (valid options are http, https, ws and wss)
--services string service names separated by space
--target-url string target url
--template string use a provider template (if empty defaults to apic template)
--title string title value (required)
-v, --version string version value (default "1.0.0")
--wsdl string wsdl file to use as the source (required authentication via apic login)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic create:product

Create a product definition

## Synopsis

Create a product definition

```
apic create:product [flags]
```

## Examples

```

Create a product
$ apic create:product --title "Climb On"
Created climb-on.yaml product definition [climb-on:1.0.0]
Create a product interactively
$ apic create:product
? Title: Climb On
? Name: climb-on
? File: climb-on.yaml
? Template:
? API Files:
Created climb-on.yaml product definition [climb-on:1.0.0]
Create a product referencing existing APIs
$ apic create:product --title "Climb On" --apis "routes.yaml ascents.yaml"
Created climb-on.yaml product definition [climb-on:1.0.0]

```

```

Create an product using one of your templates
$ apic config:set --global template-path="/etc/templates"
$ ls /etc/templates
proxy-product.hbs staging-product.hbs
$ apic create:product --title "Proxy Product" --template proxy
Created proxy-product.yaml product definition [proxy-product:1.0.0]
Create a product using your default template
$ apic config:set --global template-path="/etc/templates"
$ ls /etc/templates
proxy-product.hbs staging-product.hbs
$ apic config:set --global template-default-product=staging
$ apic create:product --title "Staging Product"
Created staging-product.yaml product definition [staging-product:1.0.0]

```

## Options

```

--apis string api file names separated by a space
--filename string filename (default derived from name)
--gateway-type string The type of the gateway (datapower-gateway, datapower-api-gateway, event-gateway) (default
"datapower-gateway")
-h, --help Help for create:product
-i, --interactive use interactive mode
--name string x-ibm-name value (default derived from title)
--template string use a provider template (if empty defaults to apic template)
--title string title value (required)
-v, --version string version value (default "1.0.0")

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic credentials

Credentials operations

## Synopsis

Credentials operations

```
apic credentials [flags]
```

## Options

```

-a, --app string Application name or id (required)
-c, --catalog string Catalog name or id (required)
--consumer-org string Consumer Organization name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for credentials
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic credentials:clear

Credentials clear operations

## Synopsis

Credentials clear operations

```
apic credentials:clear [flags]
```

## Options

---

-a, --app string	Application name or id (required)
-c, --catalog string	Catalog name or id (required)
--confirm string	Confirmation for critical updates (required)
--consumer-org string	Consumer Organization name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for credentials:clear
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic credentials:create

Create a Application Credential object

### Synopsis

---

Create a Application Credential object Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - catalog\_url - consumer\_org\_url - app\_url

```
apic credentials:create [flags]
```

### Options

---

-a, --app string	Application name or id (required)
-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for credentials:create
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic credentials:delete

Delete the Application Credential object by name or id

### Synopsis

---

Delete the Application Credential object by name or id

```
apic credentials:delete [flags]
```

### Options

---

-a, --app string	Application name or id (required)
-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for credentials:delete
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated



## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic credentials:get

Get the Application Credential object by name or id

## Synopsis

---

Get the Application Credential object by name or id

```
apic credentials:get [flags]
```

## Options

---

<code>-a, --app string</code>	Application name or id (required)
<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--consumer-org string</code>	Consumer Organization name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for credentials:get
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic credentials:list

List the Application Credential objects

## Synopsis

---

List the Application Credential objects

```
apic credentials:list [flags]
```

## Options

---

<code>-a, --app string</code>	Application name or id (required)
<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--consumer-org string</code>	Consumer Organization name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for credentials:list
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic credentials:reset-client-secret

Reset the client secret

### Synopsis

---

Reset the client secret

```
apic credentials:reset-client-secret [flags]
```

### Options

---

<code>-a, --app string</code>	Application name or id (required)
<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--consumer-org string</code>	Consumer Organization name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for credentials:reset-client-secret
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic credentials:reset

Reset the client id and client secret

### Synopsis

---

Reset the client id and client secret

```
apic credentials:reset [flags]
```

### Options

---

<code>-a, --app string</code>	Application name or id (required)
<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--consumer-org string</code>	Consumer Organization name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for credentials:reset
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic credentials:update

Update the Application Credential object by name or id

### Synopsis

---

Update the Application Credential object by name or id Fields not allowed to be null:\ - client\_id Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - catalog\_url - consumer\_org\_url - app\_url

```
apic credentials:update [flags]
```

## Options

---

-a, --app string	Application name or id (required)
-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for credentials:update
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic credentials:verify-client-secret

Verify the client secret

## Synopsis

---

Verify the client secret

```
apic credentials:verify-client-secret [flags]
```

## Options

---

-a, --app string	Application name or id (required)
-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for credentials:verify-client-secret
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic draft-apis

Draft Apis operations

## Synopsis

---

Draft Apis operations

```
apic draft-apis [flags]
```

## Options

---

--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for draft-apis
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic draft-apis:clear-all

Clear all Draft API objects in all collections

---

### Synopsis

Clear all Draft API objects in all collections

```
apic draft-apis:clear-all [flags]
```

---

### Options

```
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-apis:clear-all
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic draft-apis:clear

Clear the Draft API objects

---

### Synopsis

Clear the Draft API objects

```
apic draft-apis:clear [flags]
```

---

### Options

```
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-apis:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic draft-apis:clone

Clone the draft-api objects

---

### Synopsis

Clone the draft-api objects

```
apic draft-apis:clone [flags]
```

---

### Options

```

--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-apis:clone
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic draft-apis:create

Create a Draft API object

### Synopsis

Create a Draft API object Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url

```
apic draft-apis:create [flags]
```

### Options

```

--api_type string The type of api (asynccapi, rest, graphql, wsd1_to_rest, or wsd1)
--assembly_type string The type of the assembly to generate (rest_to_proxy)
--disable_ws_security Disable generation of WS-Security definitions in api
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
--gateway_type string The type of the gateway (datapower-gateway, datapower-api-gateway, event-gateway)
-h, --help Help for draft-apis:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--wsdl_service string Name of WSDL service to create the OpenAPI definition from

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic draft-apis:delete

Delete a Draft API

### Synopsis

Delete a Draft API

```
apic draft-apis:delete [flags]
```

### Options

```

--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-apis:delete
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

---

## apic draft-apis:document

Get the Draft API document by name and version

### Synopsis

---

Get the Draft API document by name and version

```
apic draft-apis:document [flags]
```

### Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-apis:document
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic draft-apis:get

Get the Draft API object by name and version

### Synopsis

---

Get the Draft API object by name and version

```
apic draft-apis:get [flags]
```

### Options

---

```
--fields string List of field names to return (default "add(wsdl,draft_api)")
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-apis:get
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic draft-apis:list

List the Draft API objects

### Synopsis

---

List the Draft API objects

```
apic draft-apis:list [flags]
```

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-apis:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
```

```
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic draft-apis:list-all

List all Draft API objects in all collections

### Synopsis

---

List all Draft API objects in all collections

```
apic draft-apis:list-all [flags]
```

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-apis:list-all
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic draft-apis:update

Update the Draft API object by name and version

### Synopsis

---

Update the Draft API object by name and version Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url

```
apic draft-apis:update [flags]
```

### Options

---

```
--api_type string The type of api (asynccapi, rest, graphql, wsd1_to_rest, or wsd1)
--disable_ws_security Disable generation of WS-Security definitions in api
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-apis:update
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--wsdl_service string Name of WSDL service to create the OpenAPI definition from
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic draft-apis:update-wsdl

Update with a new wsdl

## Synopsis

---

Update with a new wsdl

```
apic draft-apis:update-wsdl [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-apis:update-wsdl
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic draft-apis:validate

Validate the draft api

## Synopsis

---

Validate the draft api

```
apic draft-apis:validate [flags]
```

## Options

---

```
--catalog_name string Name of a catalog
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
--gateway_service_names string Names of Gateway Services
-h, --help Help for draft-apis:validate
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space_name string Name of a space
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic draft-apis:wsdl

Get the Draft API wsdl document by name and version

## Synopsis

---

Get the Draft API wsdl document by name and version

```
apic draft-apis:wsdl [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-apis:wsdl
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---



```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic draft-products

Draft Products operations

---

### Synopsis

Draft Products operations

```
apic draft-products [flags]
```

---

### Options

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-products
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic draft-products:clear-all

Clear all Draft Product objects in all collections

---

### Synopsis

Clear all Draft Product objects in all collections

```
apic draft-products:clear-all [flags]
```

---

### Options

```
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-products:clear-all
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic draft-products:clear

Clear the Draft Product objects

---

### Synopsis

Clear the Draft Product objects

```
apic draft-products:clear [flags]
```

## Options

---

```
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-products:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic draft-products:clone

Clone the draft-product objects

## Synopsis

---

Clone the draft-product objects

```
apic draft-products:clone [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-products:clone
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic draft-products:create

Create a Draft Product object

## Synopsis

---

Create a Draft Product object Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url

```
apic draft-products:create [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-products:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--product-only Upload only the product document
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic draft-products:delete

## Synopsis

---

Delete a Draft Product

```
apic draft-products:delete [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-products:delete
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic draft-products:document

---

Get the Draft Product document by name and version

## Synopsis

---

Get the Draft Product document by name and version

```
apic draft-products:document [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-products:document
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic draft-products:get

---

Get the Draft Product object by name and version

## Synopsis

---

Get the Draft Product object by name and version

```
apic draft-products:get [flags]
```

## Options

---

```
--fields string List of field names to return (default "add(draft_product,url)")
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-products:get
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic draft-products:list-all

List all Draft Product objects in all collections

---

### Synopsis

List all Draft Product objects in all collections

```
apic draft-products:list-all [flags]
```

---

### Options

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-products:list-all
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic draft-products:list

List the Draft Product objects

---

### Synopsis

List the Draft Product objects

```
apic draft-products:list [flags]
```

---

### Options

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for draft-products:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic draft-products:publish

Publish a draft product

---

### Synopsis

Publish a draft product

```
apic draft-products:publish [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
--gateway_services string	The list of gateway service names to support partial publishing
-h, --help	Help for draft-products:publish
--migrate_subscriptions	Migrate subscription when republish product
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--stage	stage

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic draft-products:update

Update the Draft Product object by name and version

## Synopsis

---

Update the Draft Product object by name and version Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url

```
apic draft-products:update [flags]
```

## Options

---

--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for draft-products:update
--id	id
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic draft-products:validate

Validate the draft product

## Synopsis

---

Validate the draft product

```
apic draft-products:validate [flags]
```

## Options

---

--catalog_name string	Name of a catalog
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
--gateway_service_names string	Names of Gateway Services
-h, --help	Help for draft-products:validate
--id	id
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space_name string	Name of a space
--validate_apis	Whether to validate APIs also

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic drafts

Drafts operations

---

### Synopsis

Drafts operations

```
apic drafts [flags]
```

---

### Options

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for drafts
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic drafts:clear

Clear the Draft objects

---

### Synopsis

Clear the Draft objects

```
apic drafts:clear [flags]
```

---

### Options

```
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for drafts:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic drafts:list

List the Draft objects

---

### Synopsis

List the Draft objects

```
apic drafts:list [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for drafts:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic entries

Entries operations

## Synopsis

---

Entries operations

```
apic entries [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for entries
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--truststore string Truststore name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic entries:clear

Entries clear operations

## Synopsis

---

Entries clear operations

```
apic entries:clear [flags]
```

## Options

---

```
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for entries:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--truststore string Truststore name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
```

<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic entries:create

Entries create operations

---

### Synopsis

Entries create operations

```
apic entries:create [flags]
```

---

### Options

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for entries:create
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--truststore string</code>	Truststore name or id (required)

---

### Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic entries:delete

Entries delete operations

---

### Synopsis

Entries delete operations

```
apic entries:delete [flags]
```

---

### Options

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for entries:delete
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--truststore string</code>	Truststore name or id (required)

---

### Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic entries:get

Entries get operations

---

### Synopsis

Entries get operations

```
apic entries:get [flags]
```

---

### Options



<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for entries:get
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--truststore string</code>	Truststore name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic entries:list

Entries list operations

### Synopsis

---

Entries list operations

```
apic entries:list [flags]
```

### Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for entries:list
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--truststore string</code>	Truststore name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic entries:update

Entries update operations

### Synopsis

---

Entries update operations

```
apic entries:update [flags]
```

### Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for entries:update
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--truststore string</code>	Truststore name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic extensions

Extensions operations

### Synopsis

---

Extensions operations

```
apic extensions [flags]
```

### Options

---

```
--availability-zone string Availability Zone name or id (required)
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
--gateway-service string Gateway Service name or id (required)
-h, --help Help for extensions
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic extensions:clear-all

Extensions clear-all operations

### Synopsis

---

Extensions clear-all operations

```
apic extensions:clear-all [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for extensions:clear-all
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic extensions:clear

Extensions clear operations

### Synopsis

---

Extensions clear operations

```
apic extensions:clear [flags]
```

## Options

-c, --catalog string	Catalog name or id (required)
--configured-gateway-service string	Configured Gateway Service name or id (required)
--confirm string	Confirmation for critical updates (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for extensions:clear
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic extensions:clone

Extensions clone operations

## Synopsis

Extensions clone operations

```
apic extensions:clone [flags]
```

## Options

--availability-zone string	Availability Zone name or id (required)
-c, --catalog string	Catalog name or id (required)
--configured-gateway-service string	Configured Gateway Service name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
--gateway-service string	Gateway Service name or id (required)
-h, --help	Help for extensions:clone
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic extensions:create

Extensions create operations

## Synopsis

Extensions create operations

```
apic extensions:create [flags]
```

## Options

-c, --catalog string	Catalog name or id (required)
--configured-gateway-service string	Configured Gateway Service name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to

```

yaml.
-h, --help Help for extensions:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic extensions:delete

Extensions delete operations

## Synopsis

Extensions delete operations

```
apic extensions:delete [flags]
```

## Options

```

-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for extensions:delete
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic extensions:document

Extensions document operations

## Synopsis

Extensions document operations

```
apic extensions:document [flags]
```

## Options

```

--availability-zone string Availability Zone name or id (required)
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
--gateway-service string Gateway Service name or id (required)
-h, --help Help for extensions:document
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output

```

```
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic extensions:get

Extensions get operations

---

### Synopsis

Extensions get operations

```
apic extensions:get [flags]
```

---

### Options

```
--availability-zone string Availability Zone name or id (required)
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
--gateway-service string Gateway Service name or id (required)
-h, --help Help for extensions:get
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic extensions:list

Extensions list operations

---

### Synopsis

Extensions list operations

```
apic extensions:list [flags]
```

---

### Options

```
--availability-zone string Availability Zone name or id (required)
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
--gateway-service string Gateway Service name or id (required)
-h, --help Help for extensions:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic extensions:list-all

## Synopsis

Extensions list-all operations

```
apic extensions:list-all [flags]
```

## Options

```

--availability-zone string Availability Zone name or id (required)
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
--gateway-service string Gateway Service name or id (required)
-h, --help Help for extensions:list-all
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic extensions:update

Extensions update operations

## Synopsis

Extensions update operations

```
apic extensions:update [flags]
```

## Options

```

-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for extensions:update
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic gateway-extensions

Gateway Extensions operations

## Synopsis

Gateway Extensions operations

```
apic gateway-extensions [flags]
```

## Options

---

`-h, --help` Help for gateway-extensions

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic gateway-extensions:create

Gateway Extensions create operations

## Synopsis

---

Gateway Extensions create operations

`apic gateway-extensions:create [flags]`

## Options

---

<code>--availability-zone string</code>	Availability Zone name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>--gateway-service string</code>	Gateway Service name or id (required)
<code>-h, --help</code>	Help for gateway-extensions:create
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic gateway-extensions:delete

Gateway Extensions delete operations

## Synopsis

---

Gateway Extensions delete operations

`apic gateway-extensions:delete [flags]`

## Options

---

<code>--availability-zone string</code>	Availability Zone name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>--gateway-service string</code>	Gateway Service name or id (required)
<code>-h, --help</code>	Help for gateway-extensions:delete
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic gateway-extensions:get

## Synopsis

Gateway Extensions get operations

```
apic gateway-extensions:get [flags]
```

## Options

<code>--availability-zone string</code>	Availability Zone name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>--gateway-service string</code>	Gateway Service name or id (required)
<code>-h, --help</code>	Help for gateway-extensions:get
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic gateway-extensions:implementation

Get the Gateway Extension implementation document

## Synopsis

Get the Gateway Extension implementation document

```
apic gateway-extensions:implementation [flags]
```

## Options

<code>--availability-zone string</code>	Availability Zone name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>--gateway-service string</code>	Gateway Service name or id (required)
<code>-h, --help</code>	Help for gateway-extensions:implementation
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic gateway-extensions:update

Gateway Extensions update operations

## Synopsis

Gateway Extensions update operations

```
apic gateway-extensions:update [flags]
```

## Options

<code>--availability-zone string</code>	Availability Zone name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>--gateway-service string</code>	Gateway Service name or id (required)
<code>-h, --help</code>	Help for gateway-extensions:update
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)



## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic gateway-services

Gateway Services operations

### Synopsis

---

Gateway Services operations

```
apic gateway-services [flags]
```

### Options

---

<code>--availability-zone string</code>	Availability Zone name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for gateway-services
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic gateway-services:clear

Clear the Gateway Service objects

### Synopsis

---

Clear the Gateway Service objects

```
apic gateway-services:clear [flags]
```

### Options

---

<code>--availability-zone string</code>	Availability Zone name or id (required)
<code>--confirm string</code>	Confirmation for critical updates (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for gateway-services:clear
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic gateway-services:create

Create a Gateway Service object

### Synopsis

---

Create a Gateway Service object Required fields:\ - endpoint - api\_endpoint\_base - sni Fields not allowed:\ - owned Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - availability\_zone\_url

```
apic gateway-services:create [flags]
```

## Options

---

```
--availability-zone string Availability Zone name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for gateway-services:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic gateway-services:delete

Delete the Gateway Service object by name or id

## Synopsis

---

Delete the Gateway Service object by name or id

```
apic gateway-services:delete [flags]
```

## Options

---

```
--availability-zone string Availability Zone name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for gateway-services:delete
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic gateway-services:get

Get the Gateway Service object by name or id

## Synopsis

---

Get the Gateway Service object by name or id

```
apic gateway-services:get [flags]
```

## Options

---

```
--availability-zone string Availability Zone name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for gateway-services:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
```

```
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic gateway-services:list

List the Gateway Service objects

---

### Synopsis

List the Gateway Service objects

```
apic gateway-services:list [flags]
```

---

### Options

```
--availability-zone string Availability Zone name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for gateway-services:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic gateway-services:reset-oauth-secret

Reset the oauth shared crypto material

---

### Synopsis

Reset the oauth shared crypto material

```
apic gateway-services:reset-oauth-secret [flags]
```

---

### Options

```
--availability-zone string Availability Zone name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for gateway-services:reset-oauth-secret
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic gateway-services:update

Update the Gateway Service object by name or id

---

### Synopsis

Update the Gateway Service object by name or id Fields not allowed to be null:\ - endpoint - api\_endpoint\_base - sni Fields not allowed:\ - owned - endpoint - integration\_url - gateway\_service\_type - api\_endpoint\_base Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - availability\_zone\_url

```
apic gateway-services:update [flags]
```

## Options

---

```
--availability-zone string Availability Zone name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for gateway-services:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic global-policies

---

Global Policies operations

## Synopsis

---

Global Policies operations

```
apic global-policies [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for global-policies
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic global-policies:clear

---

Global Policies clear operations

## Synopsis

---

Global Policies clear operations

```
apic global-policies:clear [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for global-policies:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic global-policies:clear-all

Global Policies clear-all operations

### Synopsis

Global Policies clear-all operations

```
apic global-policies:clear-all [flags]
```

### Options

```
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for global-policies:clear-all
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic global-policies:create

Global Policies create operations

### Synopsis

Global Policies create operations

```
apic global-policies:create [flags]
```

### Options

```
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for global-policies:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic global-policies:delete

Global Policies delete operations

## Synopsis

---

Global Policies delete operations

```
apic global-policies:delete [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--configured-gateway-service string</code>	Configured Gateway Service name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.	
<code>-h, --help</code>	Help for global-policies:delete
<code>--id</code>	id
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic global-policies:document

---

Global Policies document operations

## Synopsis

---

Global Policies document operations

```
apic global-policies:document [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--configured-gateway-service string</code>	Configured Gateway Service name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.	
<code>-h, --help</code>	Help for global-policies:document
<code>--id</code>	id
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic global-policies:get

---

Global Policies get operations

## Synopsis

---

Global Policies get operations

```
apic global-policies:get [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--configured-gateway-service string</code>	Configured Gateway Service name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to

```

yaml.
-h, --help Help for global-policies:get
 --id id
-o, --org string Organization name or id (required)
 --output string Write file(s) to directory, use - for STDOUT. (default: cwd)
 --scope string scope
-s, --server string management server endpoint (required)
 --space string Space name or id (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic global-policies:list

Global Policies list operations

### Synopsis

Global Policies list operations

```
apic global-policies:list [flags]
```

### Options

```

-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for global-policies:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic global-policies:list-all

Global Policies list-all operations

### Synopsis

Global Policies list-all operations

```
apic global-policies:list-all [flags]
```

### Options

```

-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for global-policies:list-all
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic global-policies:update

Global Policies update operations

### Synopsis

Global Policies update operations

```
apic global-policies:update [flags]
```

### Options

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--configured-gateway-service string</code>	Configured Gateway Service name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for global-policies:update
<code>--id</code>	id
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

### Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic global-policy-errors

Global Policy Errors operations

### Synopsis

Global Policy Errors operations

```
apic global-policy-errors [flags]
```

### Options

```
-h, --help Help for global-policy-errors
```

### Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic global-policy-errors:create

Global Policy Errors create operations

### Synopsis

Global Policy Errors create operations

```
apic global-policy-errors:create [flags]
```

### Options



-c, --catalog string	Catalog name or id (required)
--configured-gateway-service string	Configured Gateway Service name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.	
-h, --help	Help for global-policy-errors:create
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic global-policy-errors:delete

Global Policy Errors delete operations

### Synopsis

---

Global Policy Errors delete operations

```
apic global-policy-errors:delete [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--configured-gateway-service string	Configured Gateway Service name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.	
-h, --help	Help for global-policy-errors:delete
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic global-policy-errors:get

Global Policy Errors get operations

### Synopsis

---

Global Policy Errors get operations

```
apic global-policy-errors:get [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--configured-gateway-service string	Configured Gateway Service name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.	
-h, --help	Help for global-policy-errors:get
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output

```
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic global-policy-errors:update

Global Policy Errors update operations

---

### Synopsis

Global Policy Errors update operations

```
apic global-policy-errors:update [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for global-policy-errors:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic global-policy-posthooks

Global Policy Posthooks operations

---

### Synopsis

Global Policy Posthooks operations

```
apic global-policy-posthooks [flags]
```

---

### Options

```
-h, --help Help for global-policy-posthooks
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic global-policy-posthooks:create

Global Policy Posthooks create operations

---

### Synopsis

Global Policy Posthooks create operations

```
apic global-policy-posthooks:create [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
```

```

yaml.
-h, --help Help for global-policy-posthooks:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic global-policy-posthooks:delete

Global Policy Posthooks delete operations

### Synopsis

Global Policy Posthooks delete operations

```
apic global-policy-posthooks:delete [flags]
```

### Options

```

-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for global-policy-posthooks:delete
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic global-policy-posthooks:get

Global Policy Posthooks get operations

### Synopsis

Global Policy Posthooks get operations

```
apic global-policy-posthooks:get [flags]
```

### Options

```

-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for global-policy-posthooks:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

---

## apic global-policy-posthooks:update

Global Policy Posthooks update operations

### Synopsis

---

Global Policy Posthooks update operations

```
apic global-policy-posthooks:update [flags]
```

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--configured-gateway-service string</code>	Configured Gateway Service name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for global-policy-posthooks:update
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic global-policy-prehooks

Global Policy Prehooks operations

### Synopsis

---

Global Policy Prehooks operations

```
apic global-policy-prehooks [flags]
```

### Options

---

```
-h, --help Help for global-policy-prehooks
```

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic global-policy-prehooks:create

Global Policy Prehooks create operations

### Synopsis

---

Global Policy Prehooks create operations

```
apic global-policy-prehooks:create [flags]
```

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--configured-gateway-service string</code>	Configured Gateway Service name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for global-policy-prehooks:create
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope

```
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic global-policy-prehooks:delete

---

Global Policy Prehooks delete operations

### Synopsis

---

Global Policy Prehooks delete operations

```
apic global-policy-prehooks:delete [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for global-policy-prehooks:delete
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic global-policy-prehooks:get

---

Global Policy Prehooks get operations

### Synopsis

---

Global Policy Prehooks get operations

```
apic global-policy-prehooks:get [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for global-policy-prehooks:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic global-policy-prehooks:update

---

## Synopsis

Global Policy Prehooks update operations

```
apic global-policy-prehooks:update [flags]
```

## Options

```
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for global-policy-prehooks:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic groups

Groups operations

## Synopsis

Groups operations

```
apic groups [flags]
```

## Options

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for groups
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

## Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic groups:clear

Groups clear operations

## Synopsis

Groups clear operations

```
apic groups:clear [flags]
```

## Options

```
-c, --catalog string Catalog name or id (required)
--confirm string Confirmation for critical updates (required)
```

--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for groups:clear
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic groups:create

Groups create operations

### Synopsis

---

Groups create operations

```
apic groups:create [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for groups:create
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic groups:delete

Groups delete operations

### Synopsis

---

Groups delete operations

```
apic groups:delete [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for groups:delete
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic groups:get

Groups get operations

### Synopsis

---

Groups get operations

```
apic groups:get [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for groups:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic groups:list

Groups list operations

### Synopsis

---

Groups list operations

```
apic groups:list [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for groups:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic groups:update

Groups update operations

### Synopsis

---

Groups update operations

```
apic groups:update [flags]
```



## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for groups:update
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic iam-apikey

Log in to an IBM API Connect cloud Reserved Instance using an IBM Cloud API key

## Synopsis

---

Log in to an IBM API Connect cloud Reserved Instance using an IBM Cloud API key

```
apic iam-apikey [flags]
```

## Examples

---

```
Interactive iam-apikey login
$ apic iam-apikey
Enter your IBM Cloud details
? Server: mgmthost.com
? IBM Cloud API Key?: my-apikey
Logged into mgmthost.com successfully
```

```
Non-interactive
$ apic iam-apikey --apiKey <apikey> --server mgmthost.com
Logged into mgmthost.com successfully
```

## Options

---

<code>--apiKey string</code>	IBM cloud api key
<code>-h, --help</code>	Help for iam-apikey
<code>-s, --server string</code>	management server endpoint
<code>--tokenendpoint string</code>	iam server endpoint (default: iam.cloud.ibm.com)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic identity-providers

Identity Providers operations

## Synopsis

---

Identity Providers operations

```
apic identity-providers [flags]
```

## Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for identity-providers
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")

```
--scope string scope
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic identity-providers:list

Identity Providers list operations

### Synopsis

---

Identity Providers list operations

```
apic identity-providers:list [flags]
```

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for identity-providers:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic integrations

Integrations operations

### Synopsis

---

Integrations operations

```
apic integrations [flags]
```

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for integrations
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic integrations:clear

Integrations clear operations

## Synopsis

---

Integrations clear operations

```
apic integrations:clear [flags]
```

## Options

---

<code>--confirm</code> string	Confirmation for critical updates (required)
<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for integrations:clear
<code>--output</code> string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code> string	management server endpoint (required)
<code>--subcollection</code> string	subcollection

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic integrations:create

---

Integrations create operations

## Synopsis

---

Integrations create operations

```
apic integrations:create [flags]
```

## Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for integrations:create
<code>--output</code> string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code> string	management server endpoint (required)
<code>--subcollection</code> string	subcollection

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic integrations:delete

---

Integrations delete operations

## Synopsis

---

Integrations delete operations

```
apic integrations:delete [flags]
```

## Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for integrations:delete
<code>--output</code> string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code> string	management server endpoint (required)
<code>--subcollection</code> string	subcollection

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

---

## apic integrations:get

Integrations get operations

### Synopsis

---

Integrations get operations

```
apic integrations:get [flags]
```

### Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for integrations:get
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>-s, --server string</code>	management server endpoint (required)
<code>--subcollection string</code>	subcollection

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic integrations:list

Integrations list operations

### Synopsis

---

Integrations list operations

```
apic integrations:list [flags]
```

### Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for integrations:list
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--subcollection string</code>	subcollection

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic integrations:list-all

List all Integration objects in all collections

### Synopsis

---

List all Integration objects in all collections

```
apic integrations:list-all [flags]
```

### Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for integrations:list-all
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return

```
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic integrations:update

Integrations update operations

### Synopsis

---

Integrations update operations

```
apic integrations:update [flags]
```

### Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for integrations:update
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--subcollection string subcollection
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic invitations

Invitations operations

### Synopsis

---

Invitations operations

```
apic invitations [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for invitations
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic invitations:clear

Invitations clear operations

## Synopsis

---

Invitations clear operations

```
apic invitations:clear [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--confirm string	Confirmation for critical updates (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for invitations:clear
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic invitations:create

---

Invitations create operations

## Synopsis

---

Invitations create operations

```
apic invitations:create [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for invitations:create
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic invitations:delete

---

Invitations delete operations

## Synopsis

---

Invitations delete operations

```
apic invitations:delete [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for invitations:delete
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)

```
--space string Space name or id (required)
--space-initiated space-initiated
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic invitations:get

Invitations get operations

## Synopsis

---

Invitations get operations

```
apic invitations:get [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for invitations:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic invitations:list

Invitations list operations

## Synopsis

---

Invitations list operations

```
apic invitations:list [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for invitations:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic invitations:update

Invitations update operations

### Synopsis

---

Invitations update operations

```
apic invitations:update [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for invitations:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic jobs

Jobs operations

### Synopsis

---

Jobs operations

```
apic jobs [flags]
```

### Options

---

```
--billing string Billing name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for jobs
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic jobs:clear

Clear the Job objects

### Synopsis

---

Clear the Job objects

```
apic jobs:clear [flags]
```

### Options

---



```

--billing string Billing name or id (required)
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for jobs:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)

```

## Options inherited from parent commands

---

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic jobs:delete

Delete the Job object by name or id

### Synopsis

---

Delete the Job object by name or id

```
apic jobs:delete [flags]
```

### Options

---

```

--billing string Billing name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for jobs:delete
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)

```

## Options inherited from parent commands

---

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic jobs:get

Get the Job object by name or id

### Synopsis

---

Get the Job object by name or id

```
apic jobs:get [flags]
```

### Options

---

```

--billing string Billing name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for jobs:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)

```

## Options inherited from parent commands

---

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic jobs:list

List the Job objects

## Synopsis

---

List the Job objects

```
apic jobs:list [flags]
```

## Options

---

```
--billing string Billing name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for jobs:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic jobs:retry

Re-attempt blocked or failed job

## Synopsis

---

Re-attempt blocked or failed job

```
apic jobs:retry [flags]
```

## Options

---

```
--billing string Billing name or id (required)
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for jobs:retry
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic keystores

Keystores operations

## Synopsis

---

Keystores operations

```
apic keystores [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for keystores
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic keystores:clear

Clear the Keystore objects

### Synopsis

---

Clear the Keystore objects

```
apic keystores:clear [flags]
```

### Options

---

<code>--confirm string</code>	Confirmation for critical updates (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for keystores:clear
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic keystores:create

Create a Keystore object

### Synopsis

---

Create a Keystore object Required fields:\ - keystore Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url

```
apic keystores:create [flags]
```

### Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for keystores:create
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic keystores:delete

Delete the Keystore object by name or id

### Synopsis

---

Delete the Keystore object by name or id

```
apic keystores:delete [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for keystores:delete
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic keystores:get

Get the Keystore object by name or id

## Synopsis

---

Get the Keystore object by name or id

```
apic keystores:get [flags]
```

## Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for keystores:get
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic keystores:list

List the Keystore objects

## Synopsis

---

List the Keystore objects

```
apic keystores:list [flags]
```

## Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for keystores:list
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic keystores:update

Update the Keystore object by name or id

## Synopsis

---

Update the Keystore object by name or id Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url

```
apic keystores:update [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for keystores:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic licenses

Review the license for API Connect

## Synopsis

---

Review the license for API Connect

```
apic licenses [flags]
```

## Options

---

```
-h, --help Help for licenses
--non-ibm-license Display the non IBM license files.
--notices Display the notices file
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic log-spec

Log Spec operations

## Synopsis

---

Log Spec operations

```
apic log-spec [flags]
```

## Options

---

```
-h, --help Help for log-spec
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic log-spec:get

Get the Log Spec object

## Synopsis

---

Get the Log Spec object

```
apic log-spec:get [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for log-spec:get
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic log-spec:update

Update the Log Spec object

## Synopsis

---

Update the Log Spec object Fields allowed but ignored:\ - name - type - api\_version - scope - created\_at - updated\_at - url

```
apic log-spec:update [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for log-spec:update
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic login

Log in to an IBM API Connect cloud

## Synopsis

---

Log in to an IBM API Connect cloud

```
apic login [flags]
```

## Examples

---

```
Interactive login
$ apic login
Enter your API Connect credentials
? Server: mgmthost.com
? Realm: company/realm
? Username: tommy
? Password: password
Logged into mgmthost.com successfully
```

```
Non-interactive login
$ apic login --username tommy --password password --server mgmthost.com --realm company/realm
Logged into mgmthost.com successfully
```

## Options

---

```
--apiKey string apiKey
--context string context
-h, --help Help for login
-p, --password string password
-r, --realm string realm
-s, --server string management server endpoint
--sso sso
-u, --username string user name
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic logout

Log out of an IBM API Connect cloud

## Synopsis

---

Log out of an IBM API Connect cloud

```
apic logout [flags]
```

## Examples

---

```
Clear the local authentication credentials for mgmthost.com
$ apic logout --server mgmthost.com
Logged out of server mgmthost.com
```

## Options

---

```
-h, --help Help for logout
-s, --server string management server endpoint
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic mail-servers

Mail Servers operations

## Synopsis

---

Mail Servers operations

```
apic mail-servers [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for mail-servers
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
```

```
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic mail-servers:clear

Clear the Mail Server objects

---

### Synopsis

Clear the Mail Server objects

```
apic mail-servers:clear [flags]
```

---

### Options

```
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for mail-servers:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic mail-servers:create

Create a Mail Server object

---

### Synopsis

Create a Mail Server object Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url

```
apic mail-servers:create [flags]
```

---

### Options

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for mail-servers:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic mail-servers:delete

Delete the Mail Server object by name or id

---

### Synopsis

Delete the Mail Server object by name or id

```
apic mail-servers:delete [flags]
```

---

### Options

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for mail-servers:delete
-o, --org string Organization name or id (required)
```



```
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic mail-servers:get

Get the Mail Server object by name or id

---

### Synopsis

---

Get the Mail Server object by name or id

```
apic mail-servers:get [flags]
```

---

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for mail-servers:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

---

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic mail-servers:list

List the Mail Server objects

---

### Synopsis

---

List the Mail Server objects

```
apic mail-servers:list [flags]
```

---

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for mail-servers:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic mail-servers:test-connection

Test a Mail Server connection

---

### Synopsis

---

Test a Mail Server connection

```
apic mail-servers:test-connection [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for mail-servers:test-connection
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--test-config-only</code>	test-config-only

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic mail-servers:update

Update the Mail Server object by name or id

## Synopsis

---

Update the Mail Server object by name or id Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url

```
apic mail-servers:update [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for mail-servers:update
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic me

Me operations

## Synopsis

---

Me operations

```
apic me [flags]
```

## Options

---

<code>-h, --help</code>	Help for me
-------------------------	-------------

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic me:change-password

Change my password

## Synopsis

---

Change my password

```
apic me:change-password [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for me:change-password
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic me:delete

Delete the Me object

## Synopsis

---

Delete the Me object

```
apic me:delete [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for me:delete
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic me:get

Get the Me object

## Synopsis

---

Get the Me object

```
apic me:get [flags]
```

## Options

---

```
--context string user's login context admin/manager
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for me:get
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic me:update

Update the Me object

### Synopsis

---

Update the Me object Fields allowed but ignored:\ - name - type - api\_version - scope - created\_at - updated\_at - url

```
apic me:update [flags]
```

### Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for me:update
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic member-invitations

Member Invitations operations

### Synopsis

---

Member Invitations operations

```
apic member-invitations [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--consumer-org string Consumer Organization name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for member-invitations
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic member-invitations:clear

Member Invitations clear operations

### Synopsis

---

Member Invitations clear operations

```
apic member-invitations:clear [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--confirm string Confirmation for critical updates (required)
```

```

--consumer-org string Consumer Organization name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for member-invitations:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated

```

## Options inherited from parent commands

---

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic member-invitations:create

Member Invitations create operations

### Synopsis

Member Invitations create operations

```
apic member-invitations:create [flags]
```

### Options

```

-c, --catalog string Catalog name or id (required)
--consumer-org string Consumer Organization name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for member-invitations:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated

```

## Options inherited from parent commands

---

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic member-invitations:delete

Member Invitations delete operations

### Synopsis

Member Invitations delete operations

```
apic member-invitations:delete [flags]
```

### Options

```

-c, --catalog string Catalog name or id (required)
--consumer-org string Consumer Organization name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for member-invitations:delete
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated

```

## Options inherited from parent commands

---

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file

```

```
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic member-invitations:get

Member Invitations get operations

### Synopsis

---

Member Invitations get operations

```
apic member-invitations:get [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--consumer-org string Consumer Organization name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for member-invitations:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic member-invitations:list

Member Invitations list operations

### Synopsis

---

Member Invitations list operations

```
apic member-invitations:list [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--consumer-org string Consumer Organization name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for member-invitations:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic member-invitations:update

Member Invitations update operations

## Synopsis

---

Member Invitations update operations

```
apic member-invitations:update [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for member-invitations:update
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic members

---

Members operations

## Synopsis

---

Members operations

```
apic members [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for members
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic members:clear

---

Members clear operations

## Synopsis

---

Members clear operations

```
apic members:clear [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--confirm string	Confirmation for critical updates (required)
--consumer-org string	Consumer Organization name or id (required)

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for members:clear
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic members:create

Members create operations

### Synopsis

Members create operations

```
apic members:create [flags]
```

### Options

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--consumer-org string</code>	Consumer Organization name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for members:create
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic members:delete

Members delete operations

### Synopsis

Members delete operations

```
apic members:delete [flags]
```

### Options

<code>--cascade</code>	Cascade the behavior
<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--consumer-org string</code>	Consumer Organization name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for members:delete
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file



```
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic members:get

Members get operations

---

### Synopsis

Members get operations

```
apic members:get [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--consumer-org string Consumer Organization name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for members:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic members:list

Members list operations

---

### Synopsis

Members list operations

```
apic members:list [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--consumer-org string Consumer Organization name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for members:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic members:update

Members update operations

## Synopsis

---

Members update operations

```
apic members:update [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for members:update
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic notification-languages

---

Notification Languages operations

## Synopsis

---

Notification Languages operations

```
apic notification-languages [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for notification-languages
--notification-template string	Notification Template name or id (required)
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--subcollection string	subcollection

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic notification-languages:get

---

Notification Languages get operations

## Synopsis

---

Notification Languages get operations

```
apic notification-languages:get [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for notification-languages:get
--notification-template string	Notification Template name or id (required)

-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--subcollection string	subcollection

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic notification-languages:list

Notification Languages list operations

### Synopsis

---

Notification Languages list operations

```
apic notification-languages:list [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for notification-languages:list
--notification-template string	Notification Template name or id (required)
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--subcollection string	subcollection

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic notification-languages:update

Notification Languages update operations

### Synopsis

---

Notification Languages update operations

```
apic notification-languages:update [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for notification-languages:update
--notification-template string	Notification Template name or id (required)
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--subcollection string	subcollection

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic notification-styles

Notification Styles operations

### Synopsis

---

Notification Styles operations

```
apic notification-styles [flags]
```

### Options

---

```
-h, --help Help for notification-styles
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic notification-styles:get

Notification Styles get operations

### Synopsis

---

Notification Styles get operations

```
apic notification-styles:get [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for notification-styles:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic notification-styles:update

Notification Styles update operations

### Synopsis

---

Notification Styles update operations

```
apic notification-styles:update [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for notification-styles:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic notification-templates

Notification Templates operations

### Synopsis

---

Notification Templates operations

```
apic notification-templates [flags]
```

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for notification-templates
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic notification-templates:get

Notification Templates get operations

### Synopsis

---

Notification Templates get operations

```
apic notification-templates:get [flags]
```

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for notification-templates:get
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--subcollection string</code>	subcollection

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic notification-templates:list

Notification Templates list operations

## Synopsis

---

Notification Templates list operations

```
apic notification-templates:list [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for notification-templates:list
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--subcollection string</code>	subcollection

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic notification-templates:list-all

---

Notification Templates list-all operations

## Synopsis

---

Notification Templates list-all operations

```
apic notification-templates:list-all [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for notification-templates:list-all
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic notification-templates:update

---

Notification Templates update operations

## Synopsis

---

Notification Templates update operations

```
apic notification-templates:update [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for notification-templates:update

-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--subcollection string	subcollection

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic oauth-providers

Oauth Providers operations

### Synopsis

---

Oauth Providers operations

`apic oauth-providers [flags]`

### Options

---

--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for oauth-providers
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic oauth-providers:clear

Clear the Oauth Provider objects

### Synopsis

---

Clear the Oauth Provider objects

`apic oauth-providers:clear [flags]`

### Options

---

--confirm string	Confirmation for critical updates (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for oauth-providers:clear
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic oauth-providers:create

Create a Oauth Provider object

## Synopsis

---

Create a Oauth Provider object Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url

```
apic oauth-providers:create [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for oauth-providers:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic oauth-providers:delete

Delete the Oauth Provider object by name or id

## Synopsis

---

Delete the Oauth Provider object by name or id

```
apic oauth-providers:delete [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for oauth-providers:delete
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic oauth-providers:get

Get the Oauth Provider object by name or id

## Synopsis

---

Get the Oauth Provider object by name or id

```
apic oauth-providers:get [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for oauth-providers:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```



---

## apic oauth-providers:list

List the OAuth Provider objects

### Synopsis

---

List the OAuth Provider objects

```
apic oauth-providers:list [flags]
```

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for oauth-providers:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic oauth-providers:update

Update the OAuth Provider object by name or id

### Synopsis

---

Update the OAuth Provider object by name or id Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url

```
apic oauth-providers:update [flags]
```

### Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for oauth-providers:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic org-settings

Org Settings operations

### Synopsis

---

Org Settings operations

```
apic org-settings [flags]
```

### Options

---

```
-h, --help Help for org-settings
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic org-settings:get

Get the Organization Setting object

### Synopsis

---

Get the Organization Setting object

```
apic org-settings:get [flags]
```

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for org-settings:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic org-settings:update

Update the Organization Setting object

### Synopsis

---

Update the Organization Setting object Fields allowed but ignored:\ - name - type - api\_version - scope - created\_at - updated\_at - url

```
apic org-settings:update [flags]
```

### Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for org-settings:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic orgs

Orgs operations

### Synopsis

---

Orgs operations

```
apic orgs [flags]
```

### Options

---

<code>--expand</code>	string	List of transient field to expand
<code>--fields</code>	string	List of field names to return
<code>--format</code>	string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>		Help for orgs
<code>--limit</code>	int32	Maximum number of items to return
<code>--my</code>		my
<code>--offset</code>	int32	Offset item number from list to begin return
<code>--org_type</code>	string	Type of orgs to return
<code>--output</code>	string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code>	string	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>		Accept the license for API Connect
<code>--debug</code>		Enable debug output
<code>--debug-output</code>	string	Write debug output to file
<code>--live-help</code>		Enable or disable tracking of limited usage information
<code>-m, --mode</code>	string	Toolkit operation mode (default "apim")

## apic orgs:clear

Clear the Organization objects

### Synopsis

---

Clear the Organization objects

```
apic orgs:clear [flags]
```

### Options

---

<code>--confirm</code>	string	Confirmation for critical updates (required)
<code>--format</code>	string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>		Help for orgs:clear
<code>--output</code>	string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code>	string	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>		Accept the license for API Connect
<code>--debug</code>		Enable debug output
<code>--debug-output</code>	string	Write debug output to file
<code>--live-help</code>		Enable or disable tracking of limited usage information
<code>-m, --mode</code>	string	Toolkit operation mode (default "apim")

## apic orgs:create

Create an Organization object

### Synopsis

---

Create an Organization object Required fields:\ - owner\_url Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url

```
apic orgs:create [flags]
```

### Options

---

<code>--format</code>	string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>		Help for orgs:create
<code>--output</code>	string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code>	string	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>		Accept the license for API Connect
<code>--debug</code>		Enable debug output
<code>--debug-output</code>	string	Write debug output to file
<code>--live-help</code>		Enable or disable tracking of limited usage information
<code>-m, --mode</code>	string	Toolkit operation mode (default "apim")

## apic orgs:delete

Delete the Organization object by name or id

## Synopsis

---

Delete the Organization object by name or id

```
apic orgs:delete [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for orgs:delete
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic orgs:get

Get the Organization object by name or id

## Synopsis

---

Get the Organization object by name or id

```
apic orgs:get [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for orgs:get
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic orgs:list

List the Organization objects

## Synopsis

---

List the Organization objects

```
apic orgs:list [flags]
```

## Options

---

```
--expand string List of transient field to expand
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for orgs:list
--limit int32 Maximum number of items to return
--my my
--offset int32 Offset item number from list to begin return
--org_type string Type of orgs to return
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
```

```
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic orgs:transfer-owner

Transfer owner to an associate

---

### Synopsis

Transfer owner to an associate

```
apic orgs:transfer-owner [flags]
```

---

### Options

```
--cascade Cascade the behavior
--delete_old_owner Delete old owner
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for orgs:transfer-owner
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic orgs:update

Update the Organization object by name or id

---

### Synopsis

Update the Organization object by name or id Fields not allowed:\ - name - org\_type Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url

```
apic orgs:update [flags]
```

---

### Options

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for orgs:update
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic payment-methods

Payment Methods operations

---

### Synopsis

Payment Methods operations

```
apic payment-methods [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--consumer-org string Consumer Organization name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
```

-h, --help	Help for payment-methods
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic payment-methods:create

Create a Payment Method object

### Synopsis

Create a Payment Method object Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - consumer\_org\_url

```
apic payment-methods:create [flags]
```

### Options

-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for payment-methods:create
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic payment-methods:delete

Delete the Payment Method object by name or id

### Synopsis

Delete the Payment Method object by name or id

```
apic payment-methods:delete [flags]
```

### Options

-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for payment-methods:delete
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic payment-methods:get

Get the Payment Method object by name or id

### Synopsis

---

Get the Payment Method object by name or id

```
apic payment-methods:get [flags]
```

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--consumer-org string</code>	Consumer Organization name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for payment-methods:get
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic payment-methods:list

List the Payment Method objects

### Synopsis

---

List the Payment Method objects

```
apic payment-methods:list [flags]
```

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--consumer-org string</code>	Consumer Organization name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for payment-methods:list
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic payment-methods:update

Update the Payment Method object by name or id

### Synopsis

---

Update the Payment Method object by name or id Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - consumer\_org\_url

```
apic payment-methods:update [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--consumer-org string Consumer Organization name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for payment-methods:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic permissions

---

Permissions operations

## Synopsis

---

Permissions operations

```
apic permissions [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for permissions
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic permissions:get

---

Permissions get operations

## Synopsis

---

Permissions get operations

```
apic permissions:get [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for permissions:get
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
--subcollection string subcollection
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```



---

## apic permissions:list

Permissions list operations

### Synopsis

---

Permissions list operations

```
apic permissions:list [flags]
```

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for permissions:list
--limit int32 Maximum number of items to return
--my my
--offset int32 Offset item number from list to begin return
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--subcollection string subcollection
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic permissions:list-all

List all Permission objects in all collections

### Synopsis

---

List all Permission objects in all collections

```
apic permissions:list-all [flags]
```

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for permissions:list-all
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic policies

Policies operations

### Synopsis

---

Policies operations

```
apic policies [flags]
```

### Options

---

```
--availability-zone string Availability Zone name or id (required)
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
```

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>--gateway-service string</code>	Gateway Service name or id (required)
<code>-h, --help</code>	Help for policies
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic policies:clear

Policies clear operations

### Synopsis

Policies clear operations

```
apic policies:clear [flags]
```

### Options

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--configured-gateway-service string</code>	Configured Gateway Service name or id (required)
<code>--confirm string</code>	Confirmation for critical updates (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for policies:clear
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic policies:clear-all

Policies clear-all operations

### Synopsis

Policies clear-all operations

```
apic policies:clear-all [flags]
```

### Options

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--configured-gateway-service string</code>	Configured Gateway Service name or id (required)
<code>--confirm string</code>	Confirmation for critical updates (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for policies:clear-all
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic policies:clone

Policies clone operations

### Synopsis

Policies clone operations

`apic policies:clone [flags]`

### Options

<code>--availability-zone string</code>	Availability Zone name or id (required)
<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--configured-gateway-service string</code>	Configured Gateway Service name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>--gateway-service string</code>	Gateway Service name or id (required)
<code>-h, --help</code>	Help for policies:clone
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

### Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic policies:create

Policies create operations

### Synopsis

Policies create operations

`apic policies:create [flags]`

### Options

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--configured-gateway-service string</code>	Configured Gateway Service name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for policies:create
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--policy-file string</code>	Policy document override
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

### Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic policies:delete

Policies delete operations

## Synopsis

---

Policies delete operations

```
apic policies:delete [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--configured-gateway-service string</code>	Configured Gateway Service name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for policies:delete
<code>--id</code>	id
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic policies:document

---

Policies document operations

## Synopsis

---

Policies document operations

```
apic policies:document [flags]
```

## Options

---

<code>--availability-zone string</code>	Availability Zone name or id (required)
<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--configured-gateway-service string</code>	Configured Gateway Service name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>--gateway-service string</code>	Gateway Service name or id (required)
<code>-h, --help</code>	Help for policies:document
<code>--id</code>	id
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic policies:get

---

Policies get operations

## Synopsis

---

Policies get operations

```
apic policies:get [flags]
```

## Options

---

--availability-zone string	Availability Zone name or id (required)
-c, --catalog string	Catalog name or id (required)
--configured-gateway-service string	Configured Gateway Service name or id (required)
--fields string	List of field names to return (default "add(policy,implementation)")
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.	
--gateway-service string	Gateway Service name or id (required)
-h, --help	Help for policies:get
--id	id
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic policies:implementation

Policies implementation operations

### Synopsis

Policies implementation operations

apic policies:implementation [flags]

### Options

-c, --catalog string	Catalog name or id (required)
--configured-gateway-service string	Configured Gateway Service name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.	
-h, --help	Help for policies:implementation
--id	id
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic policies:list

Policies list operations

### Synopsis

Policies list operations

apic policies:list [flags]

### Options

--availability-zone string	Availability Zone name or id (required)
-c, --catalog string	Catalog name or id (required)
--configured-gateway-service string	Configured Gateway Service name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.	
--gateway-service string	Gateway Service name or id (required)
-h, --help	Help for policies:list
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")

--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic policies:list-all

Policies list-all operations

## Synopsis

Policies list-all operations

```
apic policies:list-all [flags]
```

## Options

--availability-zone string	Availability Zone name or id (required)
-c, --catalog string	Catalog name or id (required)
--configured-gateway-service string	Configured Gateway Service name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
--gateway-service string	Gateway Service name or id (required)
-h, --help	Help for policies:list-all
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic policies:update

Policies update operations

## Synopsis

Policies update operations

```
apic policies:update [flags]
```

## Options

-c, --catalog string	Catalog name or id (required)
--configured-gateway-service string	Configured Gateway Service name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for policies:update
--id	id
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file

```
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic portal-services

Portal Services operations

---

### Synopsis

Portal Services operations

```
apic portal-services [flags]
```

---

### Options

```
--availability-zone string Availability Zone name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for portal-services
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic portal-services:clear

Clear the Portal Service objects

---

### Synopsis

Clear the Portal Service objects

```
apic portal-services:clear [flags]
```

---

### Options

```
--availability-zone string Availability Zone name or id (required)
--confirm string Confirmation for critical updates (required)
--force Force the operation
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for portal-services:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic portal-services:create

Create a Portal Service object

---

### Synopsis

Create a Portal Service object Required fields:\ - endpoint - web\_endpoint\_base Fields not allowed:\ - owned Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - availability\_zone\_url

```
apic portal-services:create [flags]
```

## Options

---

<code>--availability-zone string</code>	Availability Zone name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for portal-services:create
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic portal-services:delete

Delete the Portal Service object by name or id

## Synopsis

---

Delete the Portal Service object by name or id

```
apic portal-services:delete [flags]
```

## Options

---

<code>--availability-zone string</code>	Availability Zone name or id (required)
<code>--force</code>	Force the operation
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for portal-services:delete
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic portal-services:get

Get the Portal Service object by name or id

## Synopsis

---

Get the Portal Service object by name or id

```
apic portal-services:get [flags]
```

## Options

---

<code>--availability-zone string</code>	Availability Zone name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for portal-services:get
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")



---

## apic portal-services:list

List the Portal Service objects

### Synopsis

---

List the Portal Service objects

```
apic portal-services:list [flags]
```

### Options

---

```
--availability-zone string Availability Zone name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for portal-services:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic portal-services:update-credentials

Update the Portal Service configuration

### Synopsis

---

Update the Portal Service configuration

```
apic portal-services:update-credentials [flags]
```

### Options

---

```
--availability-zone string Availability Zone name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for portal-services:update-credentials
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic portal-services:update

Update the Portal Service object by name or id

### Synopsis

---

Update the Portal Service object by name or id Fields not allowed to be null:\ - endpoint\_tls\_client\_profile\_url - web\_endpoint\_base Fields not allowed:\ - owned - endpoint Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - availability\_zone\_url

```
apic portal-services:update [flags]
```

### Options

---

```
--availability-zone string Availability Zone name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
```

-h, --help	Help for portal-services:update
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic primary-events

Primary Events operations

### Synopsis

Primary Events operations

`apic primary-events [flags]`

### Options

--availability-zone string	Availability Zone name or id (required)
-c, --catalog string	Catalog name or id (required)
--configured-gateway-service string	Configured Gateway Service name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
--gateway-service string	Gateway Service name or id (required)
-h, --help	Help for primary-events
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--portal-service string	Portal Service name or id (required)
--scope string	scope
-s, --server string	management server endpoint (required)
--state string	State for a webhook event in subscriber queue

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic primary-events:get

Primary Events get operations

### Synopsis

Primary Events get operations

`apic primary-events:get [flags]`

### Options

--availability-zone string	Availability Zone name or id (required)
-c, --catalog string	Catalog name or id (required)
--configured-gateway-service string	Configured Gateway Service name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
--gateway-service string	Gateway Service name or id (required)
-h, --help	Help for primary-events:get
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, use - for STDOUT. (default: cwd)
--portal-service string	Portal Service name or id (required)
--scope string	scope
-s, --server string	management server endpoint (required)

## Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic primary-events:list

Primary Events list operations

### Synopsis

Primary Events list operations

```
apic primary-events:list [flags]
```

### Options

--availability-zone string	Availability Zone name or id (required)
-c, --catalog string	Catalog name or id (required)
--configured-gateway-service string	Configured Gateway Service name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
--gateway-service string	Gateway Service name or id (required)
-h, --help	Help for primary-events:list
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--portal-service string	Portal Service name or id (required)
--scope string	scope
-s, --server string	management server endpoint (required)
--state string	State for a webhook event in subscriber queue

### Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic products

Products operations

### Synopsis

Products operations

```
apic products [flags]
```

### Options

-c, --catalog string	Catalog name or id (required)
--expand string	List of transient field to expand
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for products
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

### Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic products:clear

Products clear operations

### Synopsis

---

Products clear operations

```
apic products:clear [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for products:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic products:clear-all

Products clear-all operations

### Synopsis

---

Products clear-all operations

```
apic products:clear-all [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for products:clear-all
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic products:clone

Products clone operations

### Synopsis

---

Products clone operations

```
apic products:clone [flags]
```

### Options

---

```

-c, --catalog string Catalog name or id (required)
--expand string List of transient field to expand
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for products:clone
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

---

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic products:delete

Products delete operations

### Synopsis

---

Products delete operations

```
apic products:delete [flags]
```

### Options

---

```

-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for products:delete
--id string id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

---

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic products:document

Products document operations

### Synopsis

---

Products document operations

```
apic products:document [flags]
```

### Options

---

```

-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for products:document
--id string id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

---

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file

```

```
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic products:execute-migration-target

Products execute-migration-target operations

---

### Synopsis

Products execute-migration-target operations

```
apic products:execute-migration-target [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for products:execute-migration-target
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic products:get

Products get operations

---

### Synopsis

Products get operations

```
apic products:get [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return (default "add(catalog_product)")
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for products:get
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic products:list

Products list operations

---

### Synopsis

Products list operations

```
apic products:list [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--expand string List of transient field to expand
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for products:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic products:list-all

Products list-all operations

## Synopsis

---

Products list-all operations

```
apic products:list-all [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--expand string List of transient field to expand
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for products:list-all
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic products:migrate-subscriptions

Products migrate-subscriptions operations

## Synopsis

---

Products migrate-subscriptions operations

```
apic products:migrate-subscriptions [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for products:migrate-subscriptions
--id string id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic products:publish

Publish the product

## Synopsis

---

Publish the product

```
apic products:publish [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>--gateway_services string</code>	The list of gateway service names to support partial publishing
<code>-h, --help</code>	Help for products:publish
<code>--migrate_subscriptions</code>	Migrate subscription when republish product
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--stage</code>	stage

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic products:replace

Products replace operations

## Synopsis

---

Products replace operations

```
apic products:replace [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for products:replace
<code>--id</code>	id
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic products:set-migration-target

Products set-migration-target operations



## Synopsis

---

Products set-migration-target operations

```
apic products:set-migration-target [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for products:set-migration-target
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic products:supersede

Products supersede operations

## Synopsis

---

Products supersede operations

```
apic products:supersede [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for products:supersede
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic products:update

Products update operations

## Synopsis

---

Products update operations

```
apic products:update [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for products:update
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
```

```
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic products:validate

Validate the draft product object

## Synopsis

---

Validate the draft product object

```
apic products:validate [flags]
```

## Options

---

```
--catalog_name string Name of a catalog
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
--gateway_service_names string Names of Gateway Services
-h, --help Help for products:validate
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space_name string Name of a space
--validate_apis Whether to validate APIs also
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic properties

Properties operations

## Synopsis

---

Properties operations

```
apic properties [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for properties
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic properties:clear

Properties clear operations

## Synopsis

---

Properties clear operations

```
apic properties:clear [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--confirm string	Confirmation for critical updates (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for properties:clear
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic properties:create

---

Properties create operations

## Synopsis

---

Properties create operations

```
apic properties:create [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for properties:create
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic properties:delete

---

Properties delete operations

## Synopsis

---

Properties delete operations

```
apic properties:delete [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for properties:delete
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic properties:get

Properties get operations

## Synopsis

---

Properties get operations

```
apic properties:get [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for properties:get
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic properties:list

Properties list operations

## Synopsis

---

Properties list operations

```
apic properties:list [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for properties:list
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic properties:update

Properties update operations

## Synopsis

---

Properties update operations

```
apic properties:update [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for properties:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic registrations

---

Registrations operations

## Synopsis

---

Registrations operations

```
apic registrations [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for registrations
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic registrations:clear

---

Clear the Registration objects

## Synopsis

---

Clear the Registration objects

```
apic registrations:clear [flags]
```

## Options

---

```
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for registrations:clear
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
```

```
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic registrations:create

Create a Registration object

---

### Synopsis

Create a Registration object Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url

```
apic registrations:create [flags]
```

---

### Options

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for registrations:create
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic registrations:delete

Delete the Registration object by name or id

---

### Synopsis

Delete the Registration object by name or id

```
apic registrations:delete [flags]
```

---

### Options

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for registrations:delete
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic registrations:get

Get the Registration object by name or id

---

### Synopsis

Get the Registration object by name or id

```
apic registrations:get [flags]
```

---

### Options

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for registrations:get
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic registrations:list

List the Registration objects

### Synopsis

---

List the Registration objects

```
apic registrations:list [flags]
```

### Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for registrations:list
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic registrations:update

Update the Registration object by name or id

### Synopsis

---

Update the Registration object by name or id Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url

```
apic registrations:update [flags]
```

### Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for registrations:update
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic role-defaults

Role Defaults operations

### Synopsis

---

Role Defaults operations

```
apic role-defaults [flags]
```

## Options

---

-c, --catalog string	Catalog name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for role-defaults
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic role-defaults:clear

Role Defaults clear operations

## Synopsis

---

Role Defaults clear operations

```
apic role-defaults:clear [flags]
```

## Options

---

--confirm string	Confirmation for critical updates (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for role-defaults:clear
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--subcollection string	subcollection

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic role-defaults:create

Role Defaults create operations

## Synopsis

---

Role Defaults create operations

```
apic role-defaults:create [flags]
```

## Options

---

--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for role-defaults:create
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--subcollection string	subcollection

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")



---

## apic role-defaults:delete

Role Defaults delete operations

### Synopsis

---

Role Defaults delete operations

```
apic role-defaults:delete [flags]
```

### Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for role-defaults:delete
<code>--output</code> string	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope</code> string	scope
<code>-s, --server</code> string	management server endpoint (required)
<code>--subcollection</code> string	subcollection

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

---

## apic role-defaults:get

Role Defaults get operations

### Synopsis

---

Role Defaults get operations

```
apic role-defaults:get [flags]
```

### Options

---

<code>-c, --catalog</code> string	Catalog name or id (required)
<code>--fields</code> string	List of field names to return
<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for role-defaults:get
<code>-o, --org</code> string	Organization name or id (required)
<code>--output</code> string	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>--scope</code> string	scope
<code>-s, --server</code> string	management server endpoint (required)
<code>--space</code> string	Space name or id (required)
<code>--subcollection</code> string	subcollection

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

---

## apic role-defaults:list

Role Defaults list operations

### Synopsis

---

Role Defaults list operations

```
apic role-defaults:list [flags]
```

### Options

---

<code>-c, --catalog</code> string	Catalog name or id (required)
<code>--fields</code> string	List of field names to return
<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.

-h, --help	Help for role-defaults:list
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--subcollection string	subcollection

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic role-defaults:list-all

Role Defaults list-all operations

### Synopsis

Role Defaults list-all operations

```
apic role-defaults:list-all [flags]
```

### Options

-c, --catalog string	Catalog name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for role-defaults:list-all
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic role-defaults:update

Role Defaults update operations

### Synopsis

Role Defaults update operations

```
apic role-defaults:update [flags]
```

### Options

--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for role-defaults:update
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)
--subcollection string	subcollection

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic roles

Roles operations

### Synopsis

---

Roles operations

```
apic roles [flags]
```

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--consumer-org string</code>	Consumer Organization name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for roles
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic roles:clear

Roles clear operations

### Synopsis

---

Roles clear operations

```
apic roles:clear [flags]
```

### Options

---

<code>--confirm string</code>	Confirmation for critical updates (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for roles:clear
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic roles:create

Roles create operations

### Synopsis

---

Roles create operations

```
apic roles:create [flags]
```

### Options

---

```

--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for roles:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic roles:delete

Roles delete operations

## Synopsis

Roles delete operations

```
apic roles:delete [flags]
```

## Options

```

--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for roles:delete
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic roles:get

Roles get operations

## Synopsis

Roles get operations

```
apic roles:get [flags]
```

## Options

```

-c, --catalog string Catalog name or id (required)
--consumer-org string Consumer Organization name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for roles:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic roles:list

## Synopsis

---

Roles list operations

`apic roles:list [flags]`

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--consumer-org string</code>	Consumer Organization name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for roles:list
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic roles:update

Roles update operations

## Synopsis

---

Roles update operations

`apic roles:update [flags]`

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for roles:update
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic services

Services operations

## Synopsis

---

Services operations

`apic services [flags]`

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--configured-gateway-service string</code>	Configured Gateway Service name or id (required)
<code>--fields string</code>	List of field names to return

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.	
<code>-h, --help</code>	Help for services
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic services:clear

Services clear operations

### Synopsis

---

Services clear operations

`apic services:clear [flags]`

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--configured-gateway-service string</code>	Configured Gateway Service name or id (required)
<code>--confirm string</code>	Confirmation for critical updates (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.	
<code>-h, --help</code>	Help for services:clear
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic services:clear-all

Services clear-all operations

### Synopsis

---

Services clear-all operations

`apic services:clear-all [flags]`

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--configured-gateway-service string</code>	Configured Gateway Service name or id (required)
<code>--confirm string</code>	Confirmation for critical updates (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.	
<code>-h, --help</code>	Help for services:clear-all
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic services:create

Services create operations

---

### Synopsis

Services create operations

```
apic services:create [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for services:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic services:delete

Services delete operations

---

### Synopsis

Services delete operations

```
apic services:delete [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for services:delete
--id string id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic services:get

Services get operations

## Synopsis

---

Services get operations

```
apic services:get [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for services:get
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic services:list

---

Services list operations

## Synopsis

---

Services list operations

```
apic services:list [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for services:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic services:list-all

---

Services list-all operations

## Synopsis

---

Services list-all operations

```
apic services:list-all [flags]
```

## Options

---



```

-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for services:list-all
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic services:update

Services update operations

### Synopsis

Services update operations

```
apic services:update [flags]
```

### Options

```

-c, --catalog string Catalog name or id (required)
--configured-gateway-service string Configured Gateway Service name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for services:update
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated

```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic space-settings

Space Settings operations

### Synopsis

Space Settings operations

```
apic space-settings [flags]
```

### Options

```
-h, --help Help for space-settings
```

## Options inherited from parent commands

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

---

## apic space-settings:get

Get the Space Setting object

### Synopsis

---

Get the Space Setting object

```
apic space-settings:get [flags]
```

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for space-settings:get
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic space-settings:update

Update the Space Setting object

### Synopsis

---

Update the Space Setting object Fields allowed but ignored:\ - name - type - api\_version - scope - created\_at - updated\_at - url - org\_url - catalog\_url - space\_url

```
apic space-settings:update [flags]
```

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for space-settings:update
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic spaces

Spaces operations

### Synopsis

---

Spaces operations

```
apic spaces [flags]
```

### Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for spaces

<code>--limit int32</code>	Maximum number of items to return
<code>--my</code>	my
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic spaces:clear

Clear the Space objects

## Synopsis

---

Clear the Space objects

```
apic spaces:clear [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--confirm string</code>	Confirmation for critical updates (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for spaces:clear
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic spaces:create

Create a Space object

## Synopsis

---

Create a Space object Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - catalog\_url

```
apic spaces:create [flags]
```

## Options

---

<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for spaces:create
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic spaces:delete

Delete the Space object by name or id

## Synopsis

---

Delete the Space object by name or id

```
apic spaces:delete [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for spaces:delete
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic spaces:email-to-owners

Send email to owners of consumer organizations

## Synopsis

---

Send email to owners of consumer organizations, given consumer org and consumer group urls. For consumer group, email owners of all consumer orgs in the group.

```
apic spaces:email-to-owners [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for spaces:email-to-owners
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic spaces:get

Get the Space object by name or id

## Synopsis

---

Get the Space object by name or id

```
apic spaces:get [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for spaces:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
```

```
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic spaces:list

List the Space objects

---

### Synopsis

List the Space objects

```
apic spaces:list [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for spaces:list
--limit int32 Maximum number of items to return
--my my
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic spaces:transfer-owner

Transfer owner to an associate

---

### Synopsis

Transfer owner to an associate

```
apic spaces:transfer-owner [flags]
```

---

### Options

```
--cascade Cascade the behavior
-c, --catalog string Catalog name or id (required)
--delete_old_owner Delete old owner
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for spaces:transfer-owner
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic spaces:update

Update the Space object by name or id

---

### Synopsis

Update the Space object by name or id Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - catalog\_url

```
apic spaces:update [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for spaces:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic subscriber-events

Subscriber Events operations

## Synopsis

---

Subscriber Events operations

```
apic subscriber-events [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for subscriber-events
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--state string State for a webhook event in subscriber queue
--webhook string Webhook name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic subscriber-events:get

Subscriber Events get operations

## Synopsis

---

Subscriber Events get operations

```
apic subscriber-events:get [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for subscriber-events:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--webhook string Webhook name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
```

```
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic subscriber-events:list

Subscriber Events list operations

---

### Synopsis

Subscriber Events list operations

```
apic subscriber-events:list [flags]
```

---

### Options

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for subscriber-events:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--state string State for a webhook event in subscriber queue
--webhook string Webhook name or id (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic subscriptions

Subscriptions operations

---

### Synopsis

Subscriptions operations

```
apic subscriptions [flags]
```

---

### Options

```
-a, --app string Application name or id (required)
-c, --catalog string Catalog name or id (required)
--consumer-org string Consumer Organization name or id (required)
--consumer_org_url string Consumer Org Url
--expand string List of transient field to expand
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for subscriptions
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--plan string Plan Name
--product_url string Product Url
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
--space-initiated space-initiated
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic subscriptions:clear

Clear the Subscription objects

### Synopsis

---

Clear the Subscription objects

```
apic subscriptions:clear [flags]
```

### Options

---

<code>-a, --app string</code>	Application name or id (required)
<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--confirm string</code>	Confirmation for critical updates (required)
<code>--consumer-org string</code>	Consumer Organization name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for subscriptions:clear
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic subscriptions:create

Create a Subscription object

### Synopsis

---

Create a Subscription object Required fields:\ - product\_url - plan Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - catalog\_url - consumer\_org\_url - app\_url

```
apic subscriptions:create [flags]
```

### Options

---

<code>-a, --app string</code>	Application name or id (required)
<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--consumer-org string</code>	Consumer Organization name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for subscriptions:create
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic subscriptions:delete

Delete the Subscription object by name or id

### Synopsis

---

Delete the Subscription object by name or id

```
apic subscriptions:delete [flags]
```



## Options

---

-a, --app string	Application name or id (required)
-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for subscriptions:delete
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic subscriptions:get

Get the Subscription object by name or id

## Synopsis

---

Get the Subscription object by name or id

```
apic subscriptions:get [flags]
```

## Options

---

-a, --app string	Application name or id (required)
-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for subscriptions:get
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string	management server endpoint (required)
--space string	Space name or id (required)
--space-initiated	space-initiated

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic subscriptions:list

List the Subscription objects

## Synopsis

---

List the Subscription objects

```
apic subscriptions:list [flags]
```

## Options

---

-a, --app string	Application name or id (required)
-c, --catalog string	Catalog name or id (required)
--consumer-org string	Consumer Organization name or id (required)
--consumer-org-url string	Consumer Org Url
--expand string	List of transient field to expand
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for subscriptions:list
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--plan string	Plan Name

<code>--product_url string</code>	Product Url
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic subscriptions:update

Update the Subscription object by name or id

### Synopsis

Update the Subscription object by name or id Fields not allowed:\ - billing\_identifiers - plan\_title Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - catalog\_url - consumer\_org\_url - app\_url

`apic subscriptions:update [flags]`

### Options

<code>-a, --app string</code>	Application name or id (required)
<code>-c, --catalog string</code>	Catalog name or id (required)
<code>--consumer-org string</code>	Consumer Organization name or id (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for subscriptions:update
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--space string</code>	Space name or id (required)
<code>--space-initiated</code>	space-initiated

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic task-queues

Task Queues operations

### Synopsis

Task Queues operations

`apic task-queues [flags]`

### Options

<code>--apply_filter</code>	Filter tasks
<code>--cascade</code>	Cascade the behavior
<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for task-queues
<code>--kind string</code>	kind of item
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>--query string</code>	Add query to request
<code>--scope string</code>	scope
<code>-s, --server string</code>	management server endpoint (required)
<code>--state string</code>	State for a webhook event in subscriber queue
<code>--target string</code>	target for the task

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output

```
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic task-queues:get

Task Queues get operations

---

### Synopsis

Task Queues get operations

```
apic task-queues:get [flags]
```

---

### Options

```
--cascade Cascade the behavior
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for task-queues:get
--kind string kind of item
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--query string Add query to request
--scope string scope
-s, --server string management server endpoint (required)
--state string State for a webhook event in subscriber queue
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic task-queues:list

Task Queues list operations

---

### Synopsis

Task Queues list operations

```
apic task-queues:list [flags]
```

---

### Options

```
--apply_filter Filter tasks
--cascade Cascade the behavior
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for task-queues:list
--kind string kind of item
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
--output string Write file(s) to directory, instead of STDOUT (default "-")
--query string Add query to request
--scope string scope
-s, --server string management server endpoint (required)
--state string State for a webhook event in subscriber queue
--target string target for the task
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic tasks

Tasks operations

## Synopsis

---

Tasks operations

```
apic tasks [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for tasks
--limit int32 Maximum number of items to return
--my my
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--originated originated
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic tasks:get

Tasks get operations

## Synopsis

---

Tasks get operations

```
apic tasks:get [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for tasks:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic tasks:list

Tasks list operations

## Synopsis

---

Tasks list operations

```
apic tasks:list [flags]
```

## Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for tasks:list
```

```

--limit int32 Maximum number of items to return
--my my
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--originated originated
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

---

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic tasks:update

Tasks update operations

### Synopsis

---

Tasks update operations

```
apic tasks:update [flags]
```

### Options

---

```

-c, --catalog string Catalog name or id (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for tasks:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
--space string Space name or id (required)

```

## Options inherited from parent commands

---

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

## apic tls-client-profiles

Tls Client Profiles operations

### Synopsis

---

Tls Client Profiles operations

```
apic tls-client-profiles [flags]
```

### Options

---

```

--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for tls-client-profiles
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)

```

## Options inherited from parent commands

---

```

--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")

```

---

## apic tls-client-profiles:clear

Clear the TLS Client Profile objects

### Synopsis

---

Clear the TLS Client Profile objects

```
apic tls-client-profiles:clear [flags]
```

### Options

---

<code>--confirm</code> string	Confirmation for critical updates (required)
<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for <code>tls-client-profiles:clear</code>
<code>-o, --org</code> string	Organization name or id (required)
<code>--output</code> string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code> string	management server endpoint (required)

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

---

## apic tls-client-profiles:clear-all

Clear all TLS Client Profile objects in all collections

### Synopsis

---

Clear all TLS Client Profile objects in all collections

```
apic tls-client-profiles:clear-all [flags]
```

### Options

---

<code>--cascade</code>	Cascade the behavior
<code>--confirm</code> string	Confirmation for critical updates (required)
<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for <code>tls-client-profiles:clear-all</code>
<code>-o, --org</code> string	Organization name or id (required)
<code>--output</code> string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code> string	management server endpoint (required)

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

---

## apic tls-client-profiles:create

Create a TLS Client Profile object

### Synopsis

---

Create a TLS Client Profile object Fields not allowed:\ - owned Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url

```
apic tls-client-profiles:create [flags]
```

### Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for <code>tls-client-profiles:create</code>
<code>-o, --org</code> string	Organization name or id (required)
<code>--output</code> string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code> string	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic tls-client-profiles:delete

Delete a TLS Client Profile

### Synopsis

---

Delete a TLS Client Profile

```
apic tls-client-profiles:delete [flags]
```

### Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for tls-client-profiles:delete
<code>--id</code>	id
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic tls-client-profiles:get

Get the TLS Client Profile object by name and version

### Synopsis

---

Get the TLS Client Profile object by name and version

```
apic tls-client-profiles:get [flags]
```

### Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for tls-client-profiles:get
<code>--id</code>	id
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic tls-client-profiles:list

List the TLS Client Profile objects

### Synopsis

---

List the TLS Client Profile objects

```
apic tls-client-profiles:list [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for tls-client-profiles:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic tls-client-profiles:list-all

List all TLS Client Profile objects in all collections

## Synopsis

---

List all TLS Client Profile objects in all collections

```
apic tls-client-profiles:list-all [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for tls-client-profiles:list-all
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic tls-client-profiles:update

Update the TLS Client Profile object by name and version

## Synopsis

---

Update the TLS Client Profile object by name and version Fields not allowed:\ - owned Fields not allowed to be null:\ - ciphers - protocols Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url

```
apic tls-client-profiles:update [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for tls-client-profiles:update
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```



---

## apic tls-server-profiles

Tls Server Profiles operations

### Synopsis

---

Tls Server Profiles operations

```
apic tls-server-profiles [flags]
```

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for tls-server-profiles
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic tls-server-profiles:clear

Clear the TLS Server Profile objects

### Synopsis

---

Clear the TLS Server Profile objects

```
apic tls-server-profiles:clear [flags]
```

### Options

---

```
--cascade Cascade the behavior
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for tls-server-profiles:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic tls-server-profiles:clear-all

Clear all TLS Server Profile objects in all collections

### Synopsis

---

Clear all TLS Server Profile objects in all collections

```
apic tls-server-profiles:clear-all [flags]
```

### Options

---

```
--cascade Cascade the behavior
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for tls-server-profiles:clear-all
```

```
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic tls-server-profiles:create

Create a TLS Server Profile object

### Synopsis

---

Create a TLS Server Profile object Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url

```
apic tls-server-profiles:create [flags]
```

### Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for tls-server-profiles:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic tls-server-profiles:delete

Delete a TLS Server Profile

### Synopsis

---

Delete a TLS Server Profile

```
apic tls-server-profiles:delete [flags]
```

### Options

---

```
--cascade Cascade the behavior
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for tls-server-profiles:delete
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic tls-server-profiles:get

Get the TLS Server Profile object by name and version

### Synopsis

---

Get the TLS Server Profile object by name and version

```
apic tls-server-profiles:get [flags]
```

## Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for <code>tls-server-profiles:get</code>
<code>--id</code>	id
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic tls-server-profiles:list

List the TLS Server Profile objects

## Synopsis

---

List the TLS Server Profile objects

```
apic tls-server-profiles:list [flags]
```

## Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for <code>tls-server-profiles:list</code>
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic tls-server-profiles:list-all

List all TLS Server Profile objects in all collections

## Synopsis

---

List all TLS Server Profile objects in all collections

```
apic tls-server-profiles:list-all [flags]
```

## Options

---

<code>--fields string</code>	List of field names to return
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for <code>tls-server-profiles:list-all</code>
<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output

```
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic tls-server-profiles:update

Update the TLS Server Profile object by name and version

---

### Synopsis

Update the TLS Server Profile object by name and version Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url

```
apic tls-server-profiles:update [flags]
```

---

### Options

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for tls-server-profiles:update
--id id
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic truststores

Truststores operations

---

### Synopsis

Truststores operations

```
apic truststores [flags]
```

---

### Options

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for truststores
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic truststores:clear

Clear the Truststore objects

---

### Synopsis

Clear the Truststore objects

```
apic truststores:clear [flags]
```

---

### Options

```
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for truststores:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic truststores:create

Create a Truststore object

### Synopsis

---

Create a Truststore object Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url

```
apic truststores:create [flags]
```

### Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for truststores:create
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic truststores:delete

Delete the Truststore object by name or id

### Synopsis

---

Delete the Truststore object by name or id

```
apic truststores:delete [flags]
```

### Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for truststores:delete
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic truststores:get

Get the Truststore object by name or id

### Synopsis

---

Get the Truststore object by name or id

```
apic truststores:get [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for truststores:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic truststores:list

List the Truststore objects

## Synopsis

---

List the Truststore objects

```
apic truststores:list [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for truststores:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic truststores:update

Update the Truststore object by name or id

## Synopsis

---

Update the Truststore object by name or id Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url

```
apic truststores:update [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for truststores:update
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic user-registries

User Registries operations

### Synopsis

---

User Registries operations

```
apic user-registries [flags]
```

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for user-registries
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic user-registries:clear

Clear the User Registry objects

### Synopsis

---

Clear the User Registry objects

```
apic user-registries:clear [flags]
```

### Options

---

```
--confirm string Confirmation for critical updates (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for user-registries:clear
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic user-registries:create

Create a User Registry object

### Synopsis

---

Create a User Registry object Required fields:\ - integration\_url Fields not allowed:\ - owned Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url

```
apic user-registries:create [flags]
```

### Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for user-registries:create
-o, --org string Organization name or id (required)
```

```
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic user-registries:delete

Delete the User Registry object by name or id

### Synopsis

---

Delete the User Registry object by name or id

```
apic user-registries:delete [flags]
```

### Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for user-registries:delete
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic user-registries:execute

Execute a User Registry operation

### Synopsis

---

Execute a User Registry operation

```
apic user-registries:execute [flags]
```

### Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for user-registries:execute
--operation string Operation to perform on the user registry (get_base_dn_list, validate_password) (required)
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
--test-config-only test-config-only
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic user-registries:get

Get the User Registry object by name or id

### Synopsis

---

Get the User Registry object by name or id



```
apic user-registries:get [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for user-registries:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic user-registries:list

List the User Registry objects

## Synopsis

---

List the User Registry objects

```
apic user-registries:list [flags]
```

## Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for user-registries:list
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic user-registries:search

Search for users in the user registry

## Synopsis

---

Search for users in the user registry

```
apic user-registries:search [flags]
```

## Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for user-registries:search
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic user-registries:test-connection

Test a User Registry connection

### Synopsis

---

Test a User Registry connection

```
apic user-registries:test-connection [flags]
```

### Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for user-registries:test-connection
<code>-o, --org</code> string	Organization name or id (required)
<code>--output</code> string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code> string	management server endpoint (required)
<code>--test-config-only</code>	test-config-only

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

---

## apic user-registries:update

Update the User Registry object by name or id

### Synopsis

---

Update the User Registry object by name or id Fields not allowed:\ - owned - registry\_type - user\_registry\_managed - correlation\_data - integration\_url Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url

```
apic user-registries:update [flags]
```

### Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for user-registries:update
<code>-o, --org</code> string	Organization name or id (required)
<code>--output</code> string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code> string	management server endpoint (required)

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

---

## apic user-registry-settings

User Registry Settings operations

### Synopsis

---

User Registry Settings operations

```
apic user-registry-settings [flags]
```

### Options

---

<code>-h, --help</code>	Help for user-registry-settings
-------------------------	---------------------------------

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic user-registry-settings:get

Get the User Registry Setting object

### Synopsis

---

Get the User Registry Setting object

```
apic user-registry-settings:get [flags]
```

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for user-registry-settings:get
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic user-registry-settings:update

Update the User Registry Setting object

### Synopsis

---

Update the User Registry Setting object Fields allowed but ignored:\ - name - type - api\_version - scope - created\_at - updated\_at - url

```
apic user-registry-settings:update [flags]
```

### Options

---

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for user-registry-settings:update
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic users

Users operations

### Synopsis

---

Users operations

```
apic users [flags]
```

### Options

---

```
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for users
```

<code>--limit int32</code>	Maximum number of items to return
<code>--offset int32</code>	Offset item number from list to begin return
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--user-registry string</code>	User Registry name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic users:clear

Clear the User objects

## Synopsis

---

Clear the User objects

```
apic users:clear [flags]
```

## Options

---

<code>--confirm string</code>	Confirmation for critical updates (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for users:clear
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--user-registry string</code>	User Registry name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic users:create

Create a User object

## Synopsis

---

Create a User object Required fields:\ - username Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - user\_registry\_url

```
apic users:create [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for users:create
<code>-o, --org string</code>	Organization name or id (required)
<code>--output string</code>	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server string</code>	management server endpoint (required)
<code>--user-registry string</code>	User Registry name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic users:delete

Delete the User object by name or id

## Synopsis

---

Delete the User object by name or id

```
apic users:delete [flags]
```

## Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for users:delete
<code>-o, --org</code> string	Organization name or id (required)
<code>--output</code> string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code> string	management server endpoint (required)
<code>--user-registry</code> string	User Registry name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic users:get

Get the User object by name or id

## Synopsis

---

Get the User object by name or id

```
apic users:get [flags]
```

## Options

---

<code>--fields</code> string	List of field names to return
<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for users:get
<code>-o, --org</code> string	Organization name or id (required)
<code>--output</code> string	Write file(s) to directory, use - for STDOUT. (default: cwd)
<code>-s, --server</code> string	management server endpoint (required)
<code>--user-registry</code> string	User Registry name or id (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic users:list

List the User objects

## Synopsis

---

List the User objects

```
apic users:list [flags]
```

## Options

---

<code>--fields</code> string	List of field names to return
<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for users:list
<code>--limit</code> int32	Maximum number of items to return
<code>--offset</code> int32	Offset item number from list to begin return
<code>-o, --org</code> string	Organization name or id (required)
<code>--output</code> string	Write file(s) to directory, instead of STDOUT (default "-")
<code>-s, --server</code> string	management server endpoint (required)
<code>--user-registry</code> string	User Registry name or id (required)

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic users:request-password-reset

Send reset password link

---

### Synopsis

Send reset password link

```
apic users:request-password-reset [flags]
```

---

### Options

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for users:request-password-reset
--output string Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic users:search-provider

Users search-provider operations

---

### Synopsis

Users search-provider operations

```
apic users:search-provider [flags]
```

---

### Options

```
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for users:search-provider
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic users:update

Update the User object by name or id

---

### Synopsis

Update the User object by name or id Fields not allowed:\ - identity\_provider - username - salt - correlation\_data - force\_password\_change - last\_login\_at Fields allowed but ignored:\ - id - type - api\_version - scope - created\_at - updated\_at - url - org\_url - user\_registry\_url

```
apic users:update [flags]
```

---

### Options

--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for users:update
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string	management server endpoint (required)
--user-registry string	User Registry name or id (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic validate

Validate an API or product definition

### Synopsis

---

Validate an API or product definition

```
apic validate [FILE] [flags]
```

### Examples

---

```
Validate an API definition
$ apic validate routes.yaml
Validated routes.yaml API definition [routes:1.0]
Validate an API definition without IBM extensions
$ apic validate --no-extensions routes.yaml
Validated routes.yaml API definition [routes:1.0]
Validate a product definition and its referenced APIs
$ apic validate climb-on.yaml
Validated climb-on.yaml product definition [climb-on:1.0.0]
Validated routes.yaml API definition [valid:1.0]
Validate a product definition without validating its referenced APIs
$ apic validate --product-only climb-on.yaml
Validated climb-on.yaml product definition [climb-on:1.0.0]
```

### Options

---

-h, --help	Help for validate
--no-extensions	for API definitions, do not validate against IBM Swagger extensions
-p, --product-only	for products definitions, do not validate referenced APIs

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic version

Get the APIConnect toolkit version

### Synopsis

---

Get the APIConnect toolkit version

```
apic version [flags]
```

### Options

---

-h, --help	Help for version
------------	------------------

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic webhooks

Webhooks operations

### Synopsis

---

Webhooks operations

```
apic webhooks [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for webhooks
--limit int32 Maximum number of items to return
--offset int32 Offset item number from list to begin return
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, instead of STDOUT (default "-")
--scope string scope
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic webhooks:get

Webhooks get operations

### Synopsis

---

Webhooks get operations

```
apic webhooks:get [flags]
```

### Options

---

```
-c, --catalog string Catalog name or id (required)
--fields string List of field names to return
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for webhooks:get
-o, --org string Organization name or id (required)
--output string Write file(s) to directory, use - for STDOUT. (default: cwd)
--scope string scope
-s, --server string management server endpoint (required)
```

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic webhooks:list

Webhooks list operations

### Synopsis

---

Webhooks list operations

```
apic webhooks:list [flags]
```

### Options

---



-c, --catalog string	Catalog name or id (required)
--fields string	List of field names to return
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for webhooks:list
--limit int32	Maximum number of items to return
--offset int32	Offset item number from list to begin return
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic webhooks:update

Webhooks update operations

### Synopsis

---

Webhooks update operations

```
apic webhooks:update [flags]
```

### Options

---

-c, --catalog string	Catalog name or id (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for webhooks:update
-o, --org string	Organization name or id (required)
--output string	Write file(s) to directory, instead of STDOUT (default "-")
--scope string	scope
-s, --server string	management server endpoint (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic wsdl

Introspect WSDL files

### Synopsis

---

Introspect a WSDL file displaying all its services \$ apic wsdl WSDL\_FILE \$ apic wsdl:introspect WSDL\_FILE

```
apic wsdl [flags]
```

### Options

---

```
-h, --help Help for wsdl
```

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic wsdl:introspect

Introspect a WSDL file displaying all its services

## Synopsis

---

Introspect a WSDL file displaying all its services

```
apic wsdl:introspect WSDL_FILE [flags]
```

## Options

---

```
-h, --help Help for wsdl:introspect
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic

---

APIConnect toolkit 61bd16316ae32e02b6cd8e72d4b18e02560de16f (Built 2020-09-23T09:51:50Z)

## Synopsis

---

APIConnect toolkit 61bd16316ae32e02b6cd8e72d4b18e02560de16f (Built 2020-09-23T09:51:50Z)

```
apic [flags]
```

## Options

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
-h, --help Help for apic
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic api

---

Api operations

## Synopsis

---

Api operations

```
apic api --mode portaladmin [flags]
```

## Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for api
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic api:add-attachment

---

Add an attachment for the given api.

## Synopsis

---

Add an attachment for a api within the developer portal of the provided org and catalog. A api can hold 10 attachments. The id or name:version of a specific api needs to be provided. e.g. 'id-of-api-called-example-3' or 'example:3.0.0'.

```
apic api:add-attachment --mode portaladmin [flags]
```

## Options

---

<code>--attachment-description string</code>	A description of the attachment to be displayed to users.
<code>--attachment-name string</code>	The name given to the attachment once it has been uploaded e.g. my-file.txt (required)
<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for api:add-attachment
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic api:add-tag

Add a tag for the given api.

## Synopsis

---

Add a tag for a api within the developer portal of the provided org and catalog. The id or name:version of a specific api needs to be provided. e.g. 'id-of-api-called-example-3' or 'example:3.0.0'.

```
apic api:add-tag --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for api:add-tag
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)
<code>--tag_name string</code>	The tag name e.g. top_level_element/next_level_element (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic api:get

Get a specific api from a developer portal.

## Synopsis

---

Get a specific api from the developer portal of the provided org and catalog. The id or name:version of a specific api needs to be provided. e.g. 'id-of-api-called-example-3' or 'example:3.0.0'.

```
apic api:get --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for api:get
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic api:get-document

Get a specific entire api document from a portal.

### Synopsis

Get a specific entire api document from the portal of the provided organization and catalog. The id or name:version of a specific api needs to be provided. For example, 'id-of-api-called-example-3' or 'example:3.0.0'.

```
apic api:get-document --mode portaladmin [flags]
```

### Options

-c, --catalog string	Name or ID of the catalog that the site belongs to. (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for api:get-document
-o, --org string	Name or ID of the organization that the catalog belongs to. (required)
-s, --server string	management server endpoint (required)

### Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic api:list

Shows a list of apis from a site.

### Synopsis

Show a list of apis of the provided org and catalog.

```
apic api:list --mode portaladmin [flags]
```

### Options

-c, --catalog string	Name or ID of the catalog that the site belongs to. (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for api:list
-o, --org string	Name or ID of the organization that the catalog belongs to. (required)
-s, --server string	management server endpoint (required)

### Options inherited from parent commands

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

---

## apic api:set-icon

Set an icon image for the given api.

### Synopsis

Set an icon image for a api within the developer portal of the provided org and catalog. The existing icon will be overwritten. The id or name:version of a specific api needs to be provided. e.g. 'id-of-api-called-example-3' or 'example:3.0.0'.

```
apic api:set-icon --mode portaladmin [flags]
```

### Options

-c, --catalog string	Name or ID of the catalog that the site belongs to. (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for api:set-icon
--icon_description string	A description of the icon to be displayed to users. Used as alt text for the image. (required)

```
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic apic-config

Apic Config operations

### Synopsis

---

Apic Config operations

```
apic apic-config --mode portaladmin [flags]
```

### Options

---

```
-h, --help Help for apic-config
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic apic-config:get

Get the configuration object that the developer portal holds.

### Synopsis

---

Get the configuration object that the developer portal holds that was sent by APIM.

```
apic apic-config:get --mode portaladmin [flags]
```

### Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for apic-config:get
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic application

Application operations

### Synopsis

---

Application operations

```
apic application --mode portaladmin [flags]
```

### Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for application
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic application:get

Get a specific application from a developer portal.

## Synopsis

---

Get a specific application from the developer portal of the provided org and catalog. The id of a specific consumer organization needs to be provided.

```
apic application:get --mode portaladmin [flags]
```

## Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for application:get
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic application:list

Shows a list of applications from a site.

## Synopsis

---

Show a list of applications of the provided org and catalog.

```
apic application:list --mode portaladmin [flags]
```

## Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for application:list
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic backups

Backups operations

## Synopsis

---

Backups operations

```
apic backups --mode portaladmin [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for backups
<code>--portal_service_name string</code>	The name of the portal service (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic backups:list

List the backups present on the remote backup server.

## Synopsis

---

Lists all the backups that are currently present on the remote backup server, if one has been set.

```
apic backups:list --mode portaladmin [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for backups:list
<code>--portal_service_name string</code>	The name of the portal service (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic config

Manage configuration variables

## Synopsis

---

Manage configuration variables

```
apic config --mode portaladmin [flags]
```

## Options

---

<code>-g, --global</code>	list the global configuration variables
<code>-h, --help</code>	Help for config
<code>-l, --local</code>	list the local application configuration variables

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic config:clear

Delete all configuration variables

## Synopsis

---

Delete all configuration variables

```
apic config:clear --mode portaladmin [flags]
```

## Examples

---

```
$ apic config:clear --mode consumer
Deleted config catalog
Deleted config org
Deleted config space
```

```
$ apic config:clear --global --mode consumer
Deleted config catalog
```

## Options

---

```
-g, --global list the global configuration variables
-h, --help Help for config:clear
-l, --local list the local application configuration variables
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic config:delete

---

Delete a configuration variable

## Synopsis

---

Configuration variables:

catalog The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the --server, --organization, and --catalog command line options. The catalog URI has the form: `https://MANAGEMENT_SERVER/consumer-api/catalogs/CONSUMER_ORG_NAME/CATALOG_NAME`

org The default value of org defined by the app's or catalog's URI can be set using the org URI. The org URI has the form: `https://MANAGEMENT_SERVER/consumer-api/orgs/CONSUMER_ORG_NAME`

space The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the --server, --organization, --catalog, and --space command line options. The space URI has the form: `https://MANAGEMENT_SERVER/consumer-api/spaces/CONSUMER_ORG_NAME/CATALOG_NAME/SPACE_NAME`

cloud The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the --server and --mode command line options. The cloud URI has the form: `https://MANAGEMENT_SERVER/consumer-api/`

mode The value of mode configuration variable defines the toolkit operation mode. It can be overridden using the --mode command line option. The value can be set to apim or consumer

template-path List of places to look for handlebar templates

template-default-api Defines the default api template

template-default-product Defines the default product template

```
apic config:delete NAME... --mode portaladmin [flags]
```

## Examples

---

```
$ apic config:delete catalog --mode consumer
Deleted config catalog
```

```
$ apic config:delete org --mode consumer
Deleted config org
```

## Options

---

```
-g, --global list the global configuration variables
-h, --help Help for config:delete
-l, --local list the local application configuration variables
```



## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic config:get

---

Get a configuration variable

### Synopsis

---

Configuration variables:

**catalog** The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the `--server`, `--organization`, and `--catalog` command line options. The catalog URI has the form: `https://MANAGEMENT_SERVER/consumer-api/catalogs/CONSUMER_ORG_NAME/CATALOG_NAME`

**org** The default value of `org` defined by the app's or catalog's URI can be set using the `org` URI. The `org` URI has the form: `https://MANAGEMENT_SERVER/consumer-api/orgs/CONSUMER_ORG_NAME`

**space** The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the `--server`, `--organization`, `--catalog`, and `--space` command line options. The space URI has the form: `https://MANAGEMENT_SERVER/consumer-api/spaces/CONSUMER_ORG_NAME/CATALOG_NAME/SPACE_NAME`

**cloud** The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the `--server` and `--mode` command line options. The cloud URI has the form: `https://MANAGEMENT_SERVER/consumer-api/`

**mode** The value of `mode` configuration variable defines the toolkit operation mode. It can be overridden using the `--mode` command line option. The value can be set to `apim` or `consumer`

**template-path** List of places to look for handlebar templates

**template-default-api** Defines the default api template

**template-default-product** Defines the default product template

`apic config:get NAME --mode portaladmin [flags]`

### Examples

---

```
$ apic config:get catalog --mode consumer
catalog: https://mgmthost.com/consumer-api/catalogs/climbon/sb
```

```
$ apic config:get org --mode consumer
org: https://mgmthost.com/consumer-api/orgs/climbon
```

### Options

---

<code>-g, --global</code>	list the global configuration variables
<code>-h, --help</code>	Help for config:get
<code>-l, --local</code>	list the local application configuration variables

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic config:list

---

List the application and global configuration variables

### Synopsis

---

Configuration variables:

**catalog** The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the `--server`, `--organization`, and `--catalog` command line options. The catalog URI has the form: `https://MANAGEMENT_SERVER/consumer-api/catalogs/CONSUMER_ORG_NAME/CATALOG_NAME`

org The default value of org defined by the app's or catalog's URI can be set using the org URI. The org URI has the form: `https://MANAGEMENT_SERVER/consumer-api/orgs/CONSUMER_ORG_NAME`

space The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the `--server`, `--organization`, `--catalog`, and `--space` command line options. The space URI has the form: `https://MANAGEMENT_SERVER/consumer-api/spaces/CONSUMER_ORG_NAME/CATALOG_NAME/SPACE_NAME`

cloud The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the `--server` and `--mode` command line options. The cloud URI has the form: `https://MANAGEMENT_SERVER/consumer-api/`

mode The value of mode configuration variable defines the toolkit operation mode. It can be overridden using the `--mode` command line option. The value can be set to `apim` or `consumer`

template-path List of places to look for handlebar templates

template-default-api Defines the default api template

template-default-product Defines the default product template

```
apic config:list --mode portaladmin [flags]
```

## Examples

---

```
$ apic config --mode consumer
catalog: https://mgmthost.com/consumer-api/catalogs/climbon/sb
org: https://mgmthost.com/consumer-api/orgs/climbon
```

```
$ apic config --global --mode consumer
catalog: https://mgmthost.com/consumer-api/catalogs/climbon/sb
```

## Options

---

```
-g, --global list the global configuration variables
-h, --help Help for config:list
-l, --local list the local application configuration variables
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic config:set

---

Set or update configuration variables

## Synopsis

---

Configuration variables:

catalog The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the `--server`, `--organization`, and `--catalog` command line options. The catalog URI has the form: `https://MANAGEMENT_SERVER/consumer-api/catalogs/CONSUMER_ORG_NAME/CATALOG_NAME`

org The default value of org defined by the app's or catalog's URI can be set using the org URI. The org URI has the form: `https://MANAGEMENT_SERVER/consumer-api/orgs/CONSUMER_ORG_NAME`

space The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the `--server`, `--organization`, `--catalog`, and `--space` command line options. The space URI has the form: `https://MANAGEMENT_SERVER/consumer-api/spaces/CONSUMER_ORG_NAME/CATALOG_NAME/SPACE_NAME`

cloud The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the `--server` and `--mode` command line options. The cloud URI has the form: `https://MANAGEMENT_SERVER/consumer-api/`

mode The value of mode configuration variable defines the toolkit operation mode. It can be overridden using the `--mode` command line option. The value can be set to `apim` or `consumer`

template-path List of places to look for handlebar templates

template-default-api Defines the default api template

template-default-product Defines the default product template

```
apic config:set NAME=VALUE ... --mode portaladmin [flags]
```

## Examples

---

```
$ apic config:set --mode consumer catalog=https://mgmthost.com/consumer-api/catalogs/climbon/sb
catalog: https://mgmthost.com/consumer-api/catalogs/climbon/sb
```

```
$ apic config:set --mode consumer org=https://mgmthost2.com/consumer-api/orgs/hikeon
org: https://mgmthost2.com/consumer-api/orgs/hikeon
```

## Options

---

<code>-g, --global</code>	list the global configuration variables
<code>-h, --help</code>	Help for config:set
<code>-l, --local</code>	list the local application configuration variables

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic consumer-org

---

Consumer Org operations

## Synopsis

---

Consumer Org operations

```
apic consumer-org --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for consumer-org
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic consumer-org:get

---

Get a specific consumer org from a developer portal.

## Synopsis

---

Get a specific consumer org from the developer portal of the provided org and catalog. The id of a specific consumer organization needs to be provided.

```
apic consumer-org:get --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for consumer-org:get
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic consumer-org:list

---

Shows a list of consumer orgs from a site.

## Synopsis

---

Show a list of consumer orgs of the provided org and catalog.

```
apic consumer-org:list --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for consumer-org:list
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic content

Content operations

## Synopsis

---

Content operations

```
apic content --mode portaladmin [flags]
```

## Options

---

<code>--bundle string</code>	The bundle type for a given entity type. E.g. "page" for the "node" entity type.
<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--entity_type string</code>	Type of entity you want to perform this operation on. To find out entity/content types see the content list-types route. (required)
<code>--format string</code>	Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for content
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic content:list

Lists all of the entities in your site for the given content type and bundle.

## Synopsis

---

Lists all of the entities in your site for the given content type and bundle.

```
apic content:list --mode portaladmin [flags]
```

## Options

---

<code>--bundle string</code>	The bundle type for a given entity type. E.g. "page" for the "node" entity type.
<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--content_type string</code>	Type of content you want to perform this operation on. To find out content types see the content list-types route. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for content:list
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic content:list-types

Lists allowed content types on your site.

### Synopsis

---

Lists allowed content types on your site.

```
apic content:list-types --mode portaladmin [flags]
```

### Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for content:list-types
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic custom-module

Custom Module operations

### Synopsis

---

Custom Module operations

```
apic custom-module --mode portaladmin [flags]
```

### Options

---

<code>-h, --help</code>	Help for custom-module
-------------------------	------------------------

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic custom-module:create-export

Create a task that exports an archive that contains the custom modules of a site.

### Synopsis

---

You can create a task to export an archive of the custom modules of a site. You can then use that archive to quickly and simply override the custom modules on another site.

```
apic custom-module:create-export --mode portaladmin [flags]
```

### Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.

```
-h, --help Help for custom-module:create-export
--no-poll Do not poll the created task and just return the task ID
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic custom-module:create-import

Create a task that imports an archive that contains the custom modules of a site.

### Synopsis

---

You can create a task to import an archive of the custom modules of a site. You can use the imported archive to quickly and simply override the custom modules for the specified site.

```
apic custom-module:create-import --mode portaladmin [flags]
```

### Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for custom-module:create-import
--no-poll Do not poll the created task and just return the task ID
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic custom-module:delete-export

If running, cancels a custom module export task.

### Synopsis

---

If running, cancels the custom module export task and deletes any related artifacts that have been generated.

```
apic custom-module:delete-export --mode portaladmin [flags]
```

### Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for custom-module:delete-export
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
--task_id string ID of the task created on the queue. (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic custom-module:delete-import

If running, cancels a custom module import task.

## Synopsis

---

If running, cancels the custom module import task and deletes any related artifacts that have been generated.

```
apic custom-module:delete-import --mode portaladmin [flags]
```

## Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for custom-module:delete-import
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
--task_id string ID of the task created on the queue. (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic custom-module:get-export

When ready, streams a custom module export artifact.

## Synopsis

---

When ready, streams a custom module export artifact back to the related task id that is provided in the arguments.

```
apic custom-module:get-export --mode portaladmin [flags]
```

## Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for custom-module:get-export
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
--task_id string ID of the task created on the queue. (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic custom-module:get-export-status

Returns information about the status of this task.

## Synopsis

---

Returns information related to the task id that is provided in the arguments.

```
apic custom-module:get-export-status --mode portaladmin [flags]
```

## Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for custom-module:get-export-status
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
--task_id string ID of the task created on the queue. (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
```

<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic custom-module:get-import-status

Get the result of the custom module import task.

### Synopsis

Get the result of the custom module import task. If the import task has completed on the portal system, this command returns the result of the command. If the import task has not completed on the portal system, it returns the current status of the task.

```
apic custom-module:get-import-status --mode portaladmin [flags]
```

### Options

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for custom-module:get-import-status
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)
<code>--task_id string</code>	ID of the task created on the queue. (required)

### Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic custom-theme

Custom Theme operations

### Synopsis

Custom Theme operations

```
apic custom-theme --mode portaladmin [flags]
```

### Options

<code>-h, --help</code>	Help for custom-theme
-------------------------	-----------------------

### Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic custom-theme:create-export

Create a task that exports an archive that contains the custom themes of a site.

### Synopsis

You can create a task to export an archive of the custom themes of a site. You can then use that archive to quickly and simply override the custom themes on another site.

```
apic custom-theme:create-export --mode portaladmin [flags]
```

### Options

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for custom-theme:create-export
<code>--no-poll</code>	Do not poll the created task and just return the task ID
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)



## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic custom-theme:create-import

Create a task that imports an archive that contains the custom themes of a site.

### Synopsis

---

You can create a task to import an archive of the custom themes of a site. You can use the imported archive to quickly and simply override the custom themes for the specified site.

```
apic custom-theme:create-import --mode portaladmin [flags]
```

### Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for custom-theme:create-import
<code>--no-poll</code>	Do not poll the created task and just return the task ID
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic custom-theme:delete-export

If running, cancels a custom theme export task.

### Synopsis

---

If running, cancels the custom theme export task and deletes any related artifacts that have been generated. This does not delete any custom themes.

```
apic custom-theme:delete-export --mode portaladmin [flags]
```

### Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for custom-theme:delete-export
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)
<code>--task_id string</code>	ID of the task created on the queue. (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic custom-theme:delete-import

Cancels a running custom theme import task.

### Synopsis

---

Cancels the custom theme import task and deletes any generated artifacts.

```
apic custom-theme:delete-import --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for custom-theme:delete-import
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)
<code>--task_id string</code>	ID of the task created on the queue. (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic custom-theme:get-export

When ready, streams a custom theme export artifact.

## Synopsis

---

When ready, streams a custom theme export artifact back to the related task id that is provided in the arguments.

```
apic custom-theme:get-export --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for custom-theme:get-export
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)
<code>--task_id string</code>	ID of the task created on the queue. (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic custom-theme:get-export-status

Returns information about the status of this task.

## Synopsis

---

Returns information related to the task id that is provided in the arguments.

```
apic custom-theme:get-export-status --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for custom-theme:get-export-status
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)
<code>--task_id string</code>	ID of the task created on the queue. (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic custom-theme:get-import-status

Get the result of the custom theme import task.

## Synopsis

---

Get the result of the custom theme import task. If the import task has completed on the portal system, this command returns the result of the command. If the import task has not completed on the portal system, it returns the current status of the task.

```
apic custom-theme:get-import-status --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for custom-theme:get-import-status
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)
<code>--task_id string</code>	ID of the task created on the queue. (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic custom-translation

Custom Translation operations

## Synopsis

---

Custom Translation operations

```
apic custom-translation --mode portaladmin [flags]
```

## Options

---

<code>-h, --help</code>	Help for custom-translation
-------------------------	-----------------------------

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic custom-translation:create-export

Create a task that exports an archive that contains the custom translation of a site.

## Synopsis

---

You can create a task to export an archive of the custom translation of a site. You can then use that archive to quickly and simply override the custom translations on another site.

```
apic custom-translation:create-export --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for custom-translation:create-export
<code>--langcodes string</code>	A comma separated list of language codes e.g. "es,zh-hans"
<code>--no-poll</code>	Do not poll the created task and just return the task ID
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file

<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic custom-translation:create-import

Create a task that imports an archive that contains the custom translations of a site.

---

### Synopsis

You can create a task to import an archive of the custom translations of a site. You can use the imported archive to simply override the custom translations for the specified site. This command may take several minutes to complete.

```
apic custom-translation:create-import --mode portaladmin [flags]
```

---

### Options

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for custom-translation:create-import
<code>--no-poll</code>	Do not poll the created task and just return the task ID
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

---

### Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic custom-translation:delete-export

If running, cancels a custom translation export task.

---

### Synopsis

If running, cancels the custom translation export task and deletes any related artifacts that have been generated. This does not delete any custom translations.

```
apic custom-translation:delete-export --mode portaladmin [flags]
```

---

### Options

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for custom-translation:delete-export
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)
<code>--task_id string</code>	ID of the task created on the queue. (required)

---

### Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic custom-translation:delete-import

If running, cancels a custom translation import task.

---

### Synopsis

If running, cancels the custom translation import task and deletes any related artifacts that have been generated. This does not delete any custom translations.

```
apic custom-translation:delete-import --mode portaladmin [flags]
```

---

### Options

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.

```
-h, --help Help for custom-translation:delete-import
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
--task_id string ID of the task created on the queue. (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic custom-translation:get-export

When ready, streams a custom translation export artifact.

### Synopsis

---

When ready, streams a custom translation export artifact back to the related task id that is provided in the arguments.

```
apic custom-translation:get-export --mode portaladmin [flags]
```

### Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for custom-translation:get-export
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
--task_id string ID of the task created on the queue. (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic custom-translation:get-export-status

Returns information about the status of this task.

### Synopsis

---

Returns information related to the task id that is provided in the arguments.

```
apic custom-translation:get-export-status --mode portaladmin [flags]
```

### Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for custom-translation:get-export-status
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
--task_id string ID of the task created on the queue. (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic custom-translation:get-import-status

Get the result of the custom translation import task.

### Synopsis

---

Get the result of the custom translation import task. If the import task has completed on the portal system, this command returns the result of the command. If the import task has not completed on the portal system, it returns the current status of the task.

```
apic custom-translation:get-import-status --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for custom-translation:get-import-status
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)
<code>--task_id string</code>	ID of the task created on the queue. (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic custom-webserver-page

Custom Web Server Page operations

## Synopsis

---

Custom Web Server Page operations

```
apic custom-webserver-page --mode portaladmin [flags]
```

## Options

---

<code>-h, --help</code>	Help for custom-webserver-page
-------------------------	--------------------------------

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic custom-webserver-page:delete

Delete the custom HTML for a web server page.

## Synopsis

---

Delete the custom web server HTML content of a specific web server page (index, 404, 40x, 50x). (Will fallback to a default.)

```
apic custom-webserver-page:delete --mode portaladmin [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for custom-webserver-page:delete
<code>--page_type string</code> (required)	The page type you want to run this command against. Values: index, 404, 40x, 50x
<code>--portal_service_name string</code>	The name of the portal service (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic custom-webserver-page:get

Get the HTML for a web server page.

## Synopsis

---

Get the web server HTML content of a specific web server page (index, 404, 40x, 50x).

```
apic custom-webserver-page:get --mode portaladmin [flags]
```

## Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for custom-webserver-page:get
<code>--page_type</code> string (required)	The page type you want to run this command against. Values: index, 404, 40x, 50x
<code>--portal_service_name</code> string	The name of the portal service (required)
<code>-s, --server</code> string	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic custom-webserver-page:set

---

Set a custom HTML page for a web server page type.

## Synopsis

---

Set the custom web server HTML content of a specific web server page (index, 404, 40x, 50x). Maximum size of 8 MB.

```
apic custom-webserver-page:set --mode portaladmin [flags]
```

## Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for custom-webserver-page:set
<code>--page_type</code> string (required)	The page type you want to run this command against. Values: index, 404, 40x, 50x
<code>--portal_service_name</code> string	The name of the portal service (required)
<code>-s, --server</code> string	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic drupal-config

---

Drupal Config operations

## Synopsis

---

Drupal Config operations

```
apic drupal-config --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog</code> string	Name or ID of the catalog that the site belongs to. (required)
<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for drupal-config
<code>-o, --org</code> string	Name or ID of the organization that the catalog belongs to. (required)
<code>--prefix</code> string	The config prefix. For example, "system". No prefix will return all names in the system.
<code>-s, --server</code> string	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output

```
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic drupal-config:delete

Delete the Config object.

---

### Synopsis

Delete the Config object or a specific config key.

```
apic drupal-config:delete --mode portaladmin [flags]
```

---

### Options

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--config_name string The config object name, for example "system.site". (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for drupal-config:delete
--config_key string A config key, for example "page.front".
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic drupal-config:get

Get the Config object

---

### Synopsis

Get the Config object or a specific config key value.

```
apic drupal-config:get --mode portaladmin [flags]
```

---

### Options

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--config_key string A config key, for example "page.front".
--config_name string The config object name, for example "system.site". (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for drupal-config:get
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic drupal-config:list

List the config names by prefix

---

### Synopsis

List the config names by prefix or provide no prefix to see all.

```
apic drupal-config:list --mode portaladmin [flags]
```

---

### Options



<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for drupal-config:list
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>--prefix string</code>	The config prefix. For example, "system". No prefix will return all names in the system.
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic drupal-config:set

Set a config value.

## Synopsis

Can set a new config value or update an already existing key-value.

```
apic drupal-config:set --mode portaladmin [flags]
```

## Options

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--config_key string</code>	A config key, for example "page.front". (required)
<code>--config_name string</code>	The config object name, for example "system.site". (required)
<code>--config_value string</code>	The value to assign to the config key. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for drupal-config:set
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic drupal-state

Drupal State operations

## Synopsis

Drupal State operations

```
apic drupal-state --mode portaladmin [flags]
```

## Options

<code>-h, --help</code>	Help for drupal-state
-------------------------	-----------------------

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic drupal-state:delete

Delete the State key.

## Synopsis

Delete the specific state key.

```
apic drupal-state:delete --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for drupal-state:delete
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)
<code>--state_key string</code>	The state key, for example "system.cron_last". (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic drupal-state:get

Get the State key

## Synopsis

---

Get the State key value.

```
apic drupal-state:get --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for drupal-state:get
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)
<code>--state_key string</code>	The state key, for example "system.cron_last". (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic drupal-state:set

Set a state key value.

## Synopsis

---

Can set a new state value or update an already existing key-value.

```
apic drupal-state:set --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for drupal-state:set
<code>--input_format string</code>	The input format of the value for the state key. Values: string, integer, float, boolean, json, yaml. [default: auto]
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)
<code>--state_key string</code>	The state key, for example "system.cron_last". (required)
<code>--state_value string</code>	The value to assign to the state key. (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file

```
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic drupal-state

Drupal State operations

---

### Synopsis

Drupal State operations

```
apic drupal-state --mode portaladmin [flags]
```

---

### Options

```
-h, --help Help for drupal-state
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic drupal-state:delete

Delete the State key.

---

### Synopsis

Delete the specific state key.

```
apic drupal-state:delete --mode portaladmin [flags]
```

---

### Options

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for drupal-state:delete
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
--state_key string The state key, for example "system.cron_last". (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic drupal-state:get

Get the State key

---

### Synopsis

Get the State key value.

```
apic drupal-state:get --mode portaladmin [flags]
```

---

### Options

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for drupal-state:get
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
--state_key string The state key, for example "system.cron_last". (required)
```

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic drupal-state:set

Set a state key value.

## Synopsis

---

Can set a new state value or update an already existing key-value.

```
apic drupal-state:set --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for drupal-state:set
<code>--input_format string</code>	The input format of the value for the state key. Values: string, integer, float, boolean, json, yaml. [default: auto]
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)
<code>--state_key string</code>	The state key, for example "system.cron_last". (required)
<code>--state_value string</code>	The value to assign to the state key. (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic entity

Entity operations

## Synopsis

---

Entity operations

```
apic entity --mode portaladmin [flags]
```

## Options

---

```
-h, --help Help for entity
```

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic entity:count

Shows a count of entities from a site.

## Synopsis

---

Show a count of entities of the provided org and catalog.

```
apic entity:count --mode portaladmin [flags]
```

## Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for entity:count
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic factory-reset

Factory Reset operations

## Synopsis

---

Factory Reset operations

```
apic factory-reset --mode portaladmin [flags]
```

## Options

---

```
-h, --help Help for factory-reset
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic factory-reset:delete

Factory resets the Developer Portal.

## Synopsis

---

Performs a factory reset of the Developer Portal, deleting the portal service and all portal sites. WARNING: This command is irreversible - please ensure you have backups configured.

```
apic factory-reset:delete --mode portaladmin [flags]
```

## Options

---

```
--execute_reset string Set to true to trigger the Developer Portal reset
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
-h, --help Help for factory-reset:delete
--portal_service_endpoint string The URL endpoint of the portal service (required)
-s, --server string Management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic forums

Forums operations

## Synopsis

---

Forums operations

```
apic forums --mode portaladmin [flags]
```

## Options

---

```
-h, --help Help for forums
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic forums:disable

Disable the forum module for a given site

## Synopsis

---

Removes all forums vocabulary taxonomy terms before disabling the forum module for a given site.

```
apic forums:disable --mode portaladmin [flags]
```

## Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for forums:disable
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic forums:enable

Enable the forum module for a given site

## Synopsis

---

Enables the forum module for a given site.

```
apic forums:enable --mode portaladmin [flags]
```

## Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for forums:enable
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic ip-security-enabled

IP Security Enabled operations.

## Synopsis

---

IP Security Enabled operations. Toggle the IP security on your Developer Portal between enabled and disabled. IP security is enabled by default.

```
apic ip-security-enabled --mode portaladmin [flags]
```

## Options

---

```
-h, --help Help for ip-security-enabled
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic ip-security-enabled:update

Enable or disable IP security functionality within the Developer Portal.

## Synopsis

---

Enable or disable IP security functionality within the Developer Portal. IP security is enabled by default.

```
apic ip-security-enabled:update --mode portaladmin [flags]
```

## Options

---

```
--enabled string Set to true or false to enable or disable IP security (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for ip-security-enabled:update
--portal_service_name string The name of the portal service (required)
-s, --server string Management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic licenses

Review the license for API Connect

## Synopsis

---

Review the license for API Connect

```
apic licenses --mode portaladmin [flags]
```

## Options

---

```
-h, --help Help for licenses
--non-ibm-license Display the non IBM license files.
--notices Display the notices file
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic modules

Modules operations

## Synopsis

---

```
apic modules --mode portaladmin [flags]
```

## Options

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--core Filter out extensions that are not in drupal core.
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for modules
--no-core Filter out extensions that are provided by drupal core.
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
--package string Filter by project packages. You can use multiple comma separated values.
-s, --server string management server endpoint (required)
--status string Filter by extension status. Choices _ enabled, disabled and/or 'not installed'. You can use multiple
comma separated values.
```

## Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic modules:delete

Deletes one or more modules.

## Synopsis

Deletes one or more modules. The provided list of modules must be comma separated.

```
apic modules:delete --mode portaladmin [flags]
```

## Options

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for modules:delete
--modules string A list of modules separated by a comma. (required)
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic modules:disable

Disable one or more modules and their dependent modules.

## Synopsis

Disable one or more modules and their dependent modules. The provided list of modules must be comma separated.

```
apic modules:disable --mode portaladmin [flags]
```

## Options

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for modules:disable
--modules string A list of modules separated by a comma. (required)
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
```



```
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic modules:enable

Enable one or more module.

---

### Synopsis

Enable one or more module by providing a comma separated list.

```
apic modules:enable --mode portaladmin [flags]
```

---

### Options

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for modules:enable
--modules string A list of modules separated by a comma. (required)
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
--resolve-dependencies Attempt to download any missing dependencies. At the moment, only works when the module name is
the same as the project name.
-s, --server string management server endpoint (required)
--skip Skip automatic downloading of libraries (c.f. devel).
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic modules:list

Shows a list of available modules.

---

### Synopsis

Show a list of modules for the portal of the provided org and catalog.

```
apic modules:list --mode portaladmin [flags]
```

---

### Options

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--core Filter out extensions that are not in drupal core.
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for modules:list
--no-core Filter out extensions that are provided by drupal core.
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
--package string Filter by project packages. You can use multiple comma separated values.
-s, --server string management server endpoint (required)
--status string Filter by extension status. Choices _ enabled, disabled and/or 'not installed'. You can use multiple
comma separated values.
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic php-memory

Php Memory operations

---

### Synopsis

Php Memory operations

```
apic php-memory --mode portaladmin [flags]
```

## Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for php-memory
<code>--portal_service_name</code> string	The name of the portal service (required)
<code>-s, --server</code> string	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic php-memory:list

List the PHP memory limit on the platform.

## Synopsis

---

Lists the PHP memory\_limit setting on the portal platform.

```
apic php-memory:list --mode portaladmin [flags]
```

## Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for php-memory:list
<code>--portal_service_name</code> string	The name of the portal service (required)
<code>-s, --server</code> string	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic php-memory:update

Set the PHP memory limit on the portal platform.

## Synopsis

---

Sets the PHP memory limit on the portal platform for both admin and web containers

```
apic php-memory:update --mode portaladmin [flags]
```

## Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for php-memory:update
<code>--memory_value</code> string	The memory value in megabytes (mb) (required)
<code>--portal_service_name</code> string	The name of the portal service (required)
<code>-s, --server</code> string	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic platforms

Platforms operations

## Synopsis

---

Platforms operations

```
apic platforms --mode portaladmin [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for platforms
<code>--portal_service_name string</code>	The name of the portal service (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic platforms:list

List the platforms present on the portal service.

## Synopsis

---

Lists all the platforms that are currently present on the Portal service.

```
apic platforms:list --mode portaladmin [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for platforms:list
<code>--portal_service_name string</code>	The name of the portal service (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic product

Product operations

## Synopsis

---

Product operations

```
apic product --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for product
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic product:add-attachment

Add an attachment for the given product.

## Synopsis

Add an attachment for a product within the developer portal of the provided org and catalog. A product can hold 10 attachments. The id or name:version of a specific product needs to be provided. e.g. 'id-of-product-called-example-3' or 'example:3.0.0'.

```
apic product:add-attachment --mode portaladmin [flags]
```

## Options

<code>--attachment_description string</code>	A description of the attachment to be displayed to users.
<code>--attachment_name string</code>	The name given to the attachment once it has been uploaded e.g. my-file.txt (required)
<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for product:add-attachment
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic product:add-tag

Add a tag for the given product.

## Synopsis

Add a tag for a product within the developer portal of the provided org and catalog. The id or name:version of a specific product needs to be provided. e.g. 'id-of-product-called-example-3' or 'example:3.0.0'.

```
apic product:add-tag --mode portaladmin [flags]
```

## Options

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for product:add-tag
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)
<code>--tag_name string</code>	The tag name e.g. top_level_element/next_level_element (required)

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic product:get

Get a specific product from a developer portal.

## Synopsis

Get a specific product from the developer portal of the provided org and catalog. The id or name:version of a specific api needs to be provided. e.g. 'id-of-api-called-example-3' or 'example:3.0.0'.

```
apic product:get --mode portaladmin [flags]
```

## Options

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for product:get
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic product:get-document

Get a specific entire product document from a portal.

### Synopsis

---

Get a specific entire product document from the portal of the provided organization and catalog. The id or name:version of a specific api needs to be provided. For example, 'id-of-api-called-example-3' or 'example:3.0.0'.

```
apic product:get-document --mode portaladmin [flags]
```

### Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for product:get-document
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic product:list

Shows a list of products from a site.

### Synopsis

---

Show a list of products of the provided org and catalog.

```
apic product:list --mode portaladmin [flags]
```

### Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for product:list
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic product:set-icon

Set an icon image for the given product.

### Synopsis

---

Set an icon image for a product within the developer portal of the provided org and catalog. The existing icon will be overwritten. The id or name:version of a specific product needs to be provided. e.g. 'id-of-product-called-example-3' or 'example:3.0.0'.

```
apic product:set-icon --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for product:set-icon
<code>--icon_description string</code>	A description of the icon to be displayed to users. Used as alt text for the image. (required)
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic security

Security operations

## Synopsis

---

Security operations

```
apic security --mode portaladmin [flags]
```

## Options

---

<code>-h, --help</code>	Help for security
-------------------------	-------------------

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic security:clear-bans

Clear all user/IP bans for a site.

## Synopsis

---

Clear any existing user/IP bans.

```
apic security:clear-bans --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for security:clear-bans
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	Management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic service-ip-allowlist

Service IP Allowlist operations.

## Synopsis

---

Service IP Allowlist operations. The allowlist service enables specific IPs to be exempt from being blocked by portal security checks, for example, load balancer and proxy IPs.

```
apic service-ip-allowlist --mode portaladmin [flags]
```

## Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for service-ip-allowlist
<code>--portal_service_name</code> string	The name of the portal service (required)
<code>-s, --server</code> string	Management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic service-ip-allowlist:add

Add IP(s) to the allowlist.

## Synopsis

---

Add specific IP(s) to the allowlist.

```
apic service-ip-allowlist:add --mode portaladmin [flags]
```

## Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for service-ip-allowlist:add
<code>--ips</code> string	Comma separated list of IPs (required)
<code>--portal_service_name</code> string	The name of the portal service (required)
<code>-s, --server</code> string	Management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic service-ip-allowlist:delete

Delete all IPs from the allowlist.

## Synopsis

---

Delete all of the IPs from the allowlist.

```
apic service-ip-allowlist:delete --mode portaladmin [flags]
```

## Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for service-ip-allowlist:delete
<code>--portal_service_name</code> string	The name of the portal service (required)
<code>-s, --server</code> string	Management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic service-ip-allowlist:list

List IP(s) currently on the allowlist.

## Synopsis

---

List all of the IP(s) that are currently on the allowlist.

```
apic service-ip-allowlist:list --mode portaladmin [flags]
```

## Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for service-ip-allowlist:list
<code>--portal_service_name</code> string	The name of the portal service (required)
<code>-s, --server</code> string	Management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic service-ip-allowlist:remove

---

Remove specific IP(s) from the allowlist.

## Synopsis

---

Remove one or more specific IP(s) from the allowlist.

```
apic service-ip-allowlist:remove --mode portaladmin [flags]
```

## Options

---

<code>--format</code> string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for service-ip-allowlist:remove
<code>--ips</code> string	Comma separated list of IPs (required)
<code>--portal_service_name</code> string	The name of the portal service (required)
<code>-s, --server</code> string	Management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic site

---

Site operations

## Synopsis

---

Site operations

```
apic site --mode portaladmin [flags]
```

## Options

---

<code>-h, --help</code>	Help for site
-------------------------	---------------

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output</code> string	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode</code> string	Toolkit operation mode (default "apim")

## apic site:cache-rebuild

---



Rebuilds the cache of a specific portal site.

## Synopsis

---

Clears the existing cache and rebuilds a new one for a specific portal site.

```
apic site:cache-rebuild --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for site:cache-rebuild
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic site:check

Run a platform check against a portal site

## Synopsis

---

Performs filesystem, database and API Manager checks against a portal site. This command is useful when trying to identify platform related problems if you are having issues with your portal site.

```
apic site:check --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for site:check
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic site:login-link

Gets an admin login link for a specific Portal site.

## Synopsis

---

Returns a one time login link for the Drupal admin account for the site specified by org and catalog parameters

```
apic site:login-link --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for site:login-link
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file

<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic site:state

Obtain the current state of a portal site.

---

### Synopsis

Returns the state of a portal site. This command is useful as a general health check analysis of the portal site.

```
apic site:state --mode portaladmin [flags]
```

---

### Options

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for site:state
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

---

### Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic site-config

Site Config operations

---

### Synopsis

Site Config operations

```
apic site-config --mode portaladmin [flags]
```

---

### Options

<code>-h, --help</code>	Help for site-config
-------------------------	----------------------

---

### Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic site-config:create-export

Create a task that exports an archive that contains the configuration of a site.

---

### Synopsis

You can create a task to export an archive of the configuration of a site. You can then use that archive to quickly and simply override the configuration on another site.

```
apic site-config:create-export --mode portaladmin [flags]
```

---

### Options

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for site-config:create-export
<code>--no-poll</code>	Do not poll the created task and just return the task ID
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic site-config:create-import

Create a task that imports an archive that contains the configuration of a site.

---

### Synopsis

You can create a task to import an archive of the configuration of a site. You can use the imported archive to quickly and simply override the configuration for the specified site.

```
apic site-config:create-import --mode portaladmin [flags]
```

---

### Options

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for site-config:create-import
--no-poll Do not poll the created task and just return the task ID
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic site-config:delete-export

If running, cancels a site configuration export task.

---

### Synopsis

If running, cancels the site configuration export task and deletes any related artifacts that have been generated.

```
apic site-config:delete-export --mode portaladmin [flags]
```

---

### Options

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for site-config:delete-export
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
--task_id string ID of the task created on the queue. (required)
```

---

### Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic site-config:delete-import

Cancels a running site configuration import task.

---

### Synopsis

Cancels the site configuration import task and deletes any generated artifacts.

```
apic site-config:delete-import --mode portaladmin [flags]
```

---

### Options

-c, --catalog string	Name or ID of the catalog that the site belongs to. (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for site-config:delete-import
-o, --org string	Name or ID of the organization that the catalog belongs to. (required)
-s, --server string	management server endpoint (required)
--task_id string	ID of the task created on the queue. (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic site-config:get-export

When ready, streams a site configuration export artifact.

### Synopsis

---

When ready, streams a site configuration export artifact back to the related task id that is provided in the arguments.

```
apic site-config:get-export --mode portaladmin [flags]
```

### Options

---

-c, --catalog string	Name or ID of the catalog that the site belongs to. (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for site-config:get-export
-o, --org string	Name or ID of the organization that the catalog belongs to. (required)
-s, --server string	management server endpoint (required)
--task_id string	ID of the task created on the queue. (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic site-config:get-export-status

Returns information about the status of this task.

### Synopsis

---

Returns information related to the task id that is provided in the arguments.

```
apic site-config:get-export-status --mode portaladmin [flags]
```

### Options

---

-c, --catalog string	Name or ID of the catalog that the site belongs to. (required)
--format string	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help	Help for site-config:get-export-status
-o, --org string	Name or ID of the organization that the catalog belongs to. (required)
-s, --server string	management server endpoint (required)
--task_id string	ID of the task created on the queue. (required)

## Options inherited from parent commands

---

--accept-license	Accept the license for API Connect
--debug	Enable debug output
--debug-output string	Write debug output to file
--live-help	Enable or disable tracking of limited usage information
-m, --mode string	Toolkit operation mode (default "apim")

## apic site-config:get-import-status

Get the result of the site configuration import task.

## Synopsis

Get the result of the site configuration import task. If the import task has completed on the portal system, this command returns the result of the command. If the import task has not completed on the portal system, it returns the current status of the task.

```
apic site-config:get-import-status --mode portaladmin [flags]
```

## Options

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for site-config:get-import-status
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)
<code>--task_id string</code>	ID of the task created on the queue. (required)

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic sites

Sites operations

## Synopsis

Sites operations

```
apic sites --mode portaladmin [flags]
```

## Options

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for sites
<code>--portal_service_name string</code>	The name of the portal service (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

## apic sites:check

Run a platform check against all portal sites present on a portal service.

## Synopsis

Performs filesystem, database and API Manager checks against all portal sites on a given portal service. This command is useful when trying to identify platform related problems on one of more of your portal sites.

```
apic sites:check --mode portaladmin [flags]
```

## Options

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for sites:check
<code>--portal_service_name string</code>	The name of the portal service (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic sites:list

List the sites present on the Portal service.

## Synopsis

---

Lists all the sites that are currently present on the Portal service.

```
apic sites:list --mode portaladmin [flags]
```

## Options

---

<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for sites:list
<code>--portal_service_name string</code>	The name of the portal service (required)
<code>-s, --server string</code>	management server endpoint (required)

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic themes

Themes operations

## Synopsis

---

Themes operations

```
apic themes --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--core</code>	Filter out extensions that are not in drupal core.
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for themes
<code>--no-core</code>	Filter out extensions that are provided by drupal core.
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>--package string</code>	Filter by project packages. You can use multiple comma separated values.
<code>-s, --server string</code>	management server endpoint (required)
<code>--status string</code>	Filter by extension status. Choices <code>_ enabled, disabled and/or 'not installed'</code> . You can use multiple comma separated values.

## Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic themes:delete

Deletes one or more themes.

## Synopsis

---

Deletes one or more themes. The provided list of themes must be comma separated.

```
apic themes:delete --mode portaladmin [flags]
```

## Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for themes:delete
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)

```
-s, --server string management server endpoint (required)
--themes string A list of themes separated by a comma. (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic themes:disable

Disable one or more themes.

## Synopsis

---

Disable one or more themes and their dependent modules. The provided list of themes must be comma separated.

```
apic themes:disable --mode portaladmin [flags]
```

## Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for themes:disable
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
--themes string A list of themes separated by a comma. (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic themes:enable

Enable one or more theme.

## Synopsis

---

Enable one or more theme by providing a comma separated list.

```
apic themes:enable --mode portaladmin [flags]
```

## Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for themes:enable
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
--resolve-dependencies Attempt to download any missing dependencies. At the moment, only works when the module name is
the same as the project name.
-s, --server string management server endpoint (required)
--skip Skip automatic downloading of libraries (c.f. devel).
--themes string A list of themes separated by a comma. (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic themes:list

Shows a list of available themes.

## Synopsis

---

Show a list of themes for the portal of the provided org and catalog.

```
apic themes:list --mode portaladmin [flags]
```

## Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--core Filter out extensions that are not in drupal core.
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for themes:list
--no-core Filter out extensions that are provided by drupal core.
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
--package string Filter by project packages. You can use multiple comma separated values.
-s, --server string management server endpoint (required)
--status string Filter by extension status. Choices _ enabled, disabled and/or 'not installed'. You can use multiple
comma separated values.
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic themes:set-default

Set a portal's default theme.

## Synopsis

---

Set a portal's default theme. The theme provided must be one that is already enabled.

```
apic themes:set-default --mode portaladmin [flags]
```

## Options

---

```
-c, --catalog string Name or ID of the catalog that the site belongs to. (required)
--format string Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
-h, --help Help for themes:set-default
-o, --org string Name or ID of the organization that the catalog belongs to. (required)
-s, --server string management server endpoint (required)
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

## apic twig

Twig operations.

## Synopsis

---

Twig operations.

```
apic twig --mode portaladmin [flags]
```

## Options

---

```
-h, --help Help for twig
```

## Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```



---

## apic twig:debug-disable

Disable twig debugging on a specific portal site.

### Synopsis

---

Disable twig debugging on a specific portal site. Twig debugging helps locate twig template issues.

```
apic twig:debug-disable --mode portaladmin [flags]
```

### Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for twig:debug-disable
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic twig:debug-enable

Enables twig debugging on a specific portal site.

### Synopsis

---

Enables twig debugging on a specific portal site. Twig debugging helps locate twig template issues.

```
apic twig:debug-enable --mode portaladmin [flags]
```

### Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for twig:debug-enable
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

### Options inherited from parent commands

---

<code>--accept-license</code>	Accept the license for API Connect
<code>--debug</code>	Enable debug output
<code>--debug-output string</code>	Write debug output to file
<code>--live-help</code>	Enable or disable tracking of limited usage information
<code>-m, --mode string</code>	Toolkit operation mode (default "apim")

---

## apic twig:debug-status

Shows the current state of twig debugging on a specific portal site.

### Synopsis

---

View the current state of twig debugging on a specific portal site, if you are unsure whether it is enabled or disabled.

```
apic twig:debug-status --mode portaladmin [flags]
```

### Options

---

<code>-c, --catalog string</code>	Name or ID of the catalog that the site belongs to. (required)
<code>--format string</code>	Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
<code>-h, --help</code>	Help for twig:debug-status
<code>-o, --org string</code>	Name or ID of the organization that the catalog belongs to. (required)
<code>-s, --server string</code>	management server endpoint (required)

### Options inherited from parent commands

---

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## apic version

Get the APIConnect toolkit version

---

## Synopsis

Get the APIConnect toolkit version

```
apic version --mode portaladmin [flags]
```

---

## Options

```
-h, --help Help for version
```

---

## Options inherited from parent commands

```
--accept-license Accept the license for API Connect
--debug Enable debug output
--debug-output string Write debug output to file
--live-help Enable or disable tracking of limited usage information
-m, --mode string Toolkit operation mode (default "apim")
```

---

## API Connect REST APIs

The API Connect platform REST APIs provide complete access to the capability of the platform.

The API Connect platform REST APIs can be used for the following actions:

- Automate the administration of the platform.
- Implement scripts and tools to support a continuous integration environment for API development and publishing.
- Manage catalogs of APIs, and their subscribers.

Note: To authenticate with the API Connect Platform REST APIs, [generate an API Key](#) and then exchange it for an access token to use when calling the APIs. You can exchange it for an access token with the following example:

```
curl -v -k -X POST -d '{"api_key": "*****", "client_id": "client-id", "client_secret": "client-secret", "grant_type": "api_key"}' -H 'Content-Type: application/json' -H 'Accept: application/json' https://platform-api.{{region}}.apiconnect.automation.ibm.com/api/token
```

The operations provided in the REST APIs also correspond directly with the commands in the [developer toolkit command-line tool \(CLI\)](#).

You can use the URL to navigate directly to specific APIs and operations in the OpenAPI Explorer Documentation for the API Connect REST APIs. You can also select the APIs for a specific version by using the drop-down menu in the header bar.

For descriptions of the platform APIs, see the [API Connect REST API documentation](#).

---

## Example: Using the platform REST APIs to publish a product containing a SOAP API

Review this sample scenario to see how you can use the API Connect platform REST APIs to publish a product that contains a SOAP API.

This example demonstrates how to format a request to publish the product, and includes sample files that illustrate the product definition, the API definition, and the SOAP API's WSDL description.

---

## Using the Catalogs API

Publish the product using the API Connect platform **Catalogs** API:

- To publish a product to a catalog, use the following syntax:  

```
POST catalogs/{org}/{catalog}/publish
```
- To publish a product to a space within a catalog, additionally specify the space with the following syntax:  

```
POST catalogs/{org}/{catalog}/{space}/publish
```

The payload in both cases is a multipart, form-encoded body containing three required parts and one optional part:

- (Required) A part named **product** with media type `application/yaml` or `application/json`, containing the yaml or JSON source for the product to be published.

- (Required) One or more occurrences of a part named `openapi` with media type `application/yaml` or `application/json`, containing the YAML or JSON source for each API in the product.
- (Required for SOAP APIs) One or more occurrences of a part named `wsdl` with media type `application/wsdl`, `application/wsdl+xml`, `text/xml`, or `application/zip` containing the WSDL definitions (and referenced XSD files if applicable) associated with the API content.
- (Optional) A part named `gateway_service_urls` with media type `application/yaml` or `application/json` which is a YAML or JSON array of URLs to specify a subset of the configured gateway services within the catalog or space as publish targets. These URLs can be obtained using the following calls:
  - For a catalog:

```
GET /api/catalogs/{org}/{catalog}/configured-gateway-services
```

- For a space:

```
GET /spaces/{org}/{catalog}/{space}/configured-gateway-services
```

Within the `product` content, the form of the references to APIs in the `apis` section is significant. Each API reference must use the `name` property instead of the `$ref` form that may appear in other situations. Here is an example showing a product that refers to the API `globalweather` version `1.0.0` in the correct form:

```
info:
 version: 1.0.0
 title: GlobalWeatherProduct
 name: globalweatherproduct
plans:
 ...
apis:
 globalweather1.0.0:
 name: globalweather:1.0.0
...
```

## Example publish operation using cURL

This example publishes a product containing a SOAP API to the `Sandbox` catalog of the `onlinebanking` provider organization, using the `curl -F` option to create a set of form-encoded payload parts and the `@` prefix to read the content for each part from a file. Note the part name (`product`, `openapi`, `wsdl`) and the content type of each part given by the `type=` value.

In this example, the access token has been retrieved earlier and placed in an environment variable named `TOKEN`. The command is split onto multiple lines for ease of reading but must be entered as a single line.

Request:

```
curl -v -k
-F "product=@globalweatherproduct_1.0.0.yaml;type=application/yaml"
-F "openapi=@globalweather_1.0.0.yaml;type=application/yaml"
-F "wsdl=@globalweather.wsdl;type=application/wsdl"
-H "Authorization: Bearer $TOKEN"
-H 'Content-Type: multipart/form-data'
-H 'Accept: */*' https://dev.apic.example.com/api/catalogs/onlinebanking/sandbox/publish
```

Response:

201 Created

```
{
 "type": "product",
 "api_version": "2.0.0",
 "id": "d03129a4-9a99-4c88-85e6-d51bbf312164",
 "name": "globalweatherproduct",
 "version": "1.0.0",
 "title": "GlobalWeatherProduct",
 "state": "published",
 "scope": "catalog",
 "gateway_types": [
 "datapower-api-gateway"
],
 "billing_urls": [],
 "api_urls": [
 "https://dev.apic.example.com/api/catalogs/87227774-d297-4064-8908-b8f0b1db71a5/1bc8f4ba-4178-4a4b-93e8-96a8ca9667fd/apis/d5986b78-d556-4457-8c38-951c471f57fc"
],
 "gateway_service_urls": [
 "https://dev.apic.example.com/api/catalogs/87227774-d297-4064-8908-b8f0b1db71a5/1bc8f4ba-4178-4a4b-93e8-96a8ca9667fd/configured-gateway-services/a416f39d-39e1-484c-9f7e-4a599abefec8"
],
 "oauth_provider_urls": [],
 "plans": [
 {
 "title": "Default Plan",
 "name": "default-plan",
 "apis": [
 {
 "id": "d5986b78-d556-4457-8c38-951c471f57fc",
 "name": "globalweather",
 "title": "GlobalWeather",
 "version": "1.0.0",
 "url": "https://dev.apic.example.com/api/catalogs/87227774-d297-4064-8908-b8f0b1db71a5/1bc8f4ba-4178-4a4b-93e8-96a8ca9667fd/apis/d5986b78-d556-4457-8c38-951c471f57fc"
 }
]
 }
],
 "visibility": {
 "view": {
 "type": "public",

```

```

 "enabled": true
 },
 "subscribe": {
 "type": "authenticated",
 "enabled": true
 }
 },
 "task_urls": [],
 "created_at": "2021-12-15T07:35:40.295Z",
 "updated_at": "2021-12-15T07:35:40.295Z",
 "org_url": "https://dev.apic.example.com/api/orgs/87227774-d297-4064-8908-b8f0b1db71a5",
 "catalog_url": "https://dev.apic.example.com/api/catalogs/87227774-d297-4064-8908-b8f0b1db71a5/1bc8f4ba-4178-4a4b-93e8-96a8ca9667fd",
 "url": "https://dev.apic.example.com/api/catalogs/87227774-d297-4064-8908-b8f0b1db71a5/1bc8f4ba-4178-4a4b-93e8-96a8ca9667fd/products/d03129a4-9a99-4c88-85e6-d51bbf312164"
}

```

The request specifies three files, which are included with this example for reference:

- [Product definition](#) -F "product=@globalweatherproduct\_1.0.0.yaml;type=application/yaml"
- [API definition](#) -F "openapi=@globalweather\_1.0.0.yaml;type=application/yaml"
- [WSDL description](#) -F "wsdl=@globalweather.wsdl;type=application/wsdl"

---

## Example Product file: globalweatherproduct\_1.0.0.yaml

The following code sample contains a Product definition.

The Product definition file globalweatherproduct\_1.0.0.yaml is referenced by the example scenario in [Example: Using the platform REST APIs to publish a product containing a SOAP API](#).

```

info:
 version: 1.0.0
 title: GlobalWeatherProduct
 name: globalweatherproduct
gateways:
- datapower-api-gateway
plans:
 default-plan:
 title: Default Plan
 description: Default Plan
 rate-limits:
 default:
 value: 100/1hour
 apis:
 globalweather1.0.0: {}
apis:
 globalweather1.0.0:
 name: globalweather:1.0.0
visibility:
 view:
 type: public
 orgs: []
 tags: []
 enabled: true
 subscribe:
 type: authenticated
 orgs: []
 tags: []
 enabled: true
product: 1.0.0

```

---

## Example API definition: globalweather\_1.0.0.yaml

The following code sample contains an API definition.

The API definition file globalweather\_1.0.0.yaml is referenced by the example scenario in [Example: Using the platform REST APIs to publish a product containing a SOAP API](#).

```

swagger: '2.0'
info:
 title: GlobalWeather
 description: ''
 x-ibm-name: globalweather
 version: 1.0.0
schemes:
- https
basePath: /globalweather
produces:
- application/xml
consumes:
- text/xml
securityDefinitions:
 clientID:
 type: apiKey
 in: header
 name: X-IBM-Client-Id
security:
- clientID: []

```

```

x-ibm-configuration:
 type: wsdl
 phase: realized
 enforced: true
 testable: true
 gateway: datapower-api-gateway
 cors:
 enabled: true
 wsdl-definition:
 wsdl: globalweather.wsdl
 service: GlobalWeather
 port: GlobalWeatherSoap
 soap-version: '1.1'
 assembly:
 execute:
 - invoke:
 title: invoke
 target-url: 'http://www.webserviceX.com/globalweather.asmx'
 version: 2.0.0
 header-control:
 type: blocklist
 values: []
 parameter-control:
 type: blocklist
 values: []
 x-ibm-apiconnect-wsdl:
 package-version: 2.0.35
 options: {}
 messages:
 info:
 - message: >-
 The wsdl 'service' has multiple 'ports'. The api is generated using
 information in the first soap 'port'.
 warning: []
 error: []
 paths:
 /GetWeather:
 post:
 summary: Operation GetWeather
 description: Get weather report for all major cities around the world.
 operationId: GetWeather
 x-ibm-soap:
 soap-action: 'http://www.webserviceX.NET/GetWeather'
 soap-operation: '{http://www.webserviceX.NET}GetWeather'
 parameters:
 - in: body
 name: body
 required: true
 schema:
 $ref: '#/definitions/GetWeatherInput'
 responses:
 default:
 description: ''
 schema:
 $ref: '#/definitions/GetWeatherOutput'
 /GetCitiesByCountry:
 post:
 summary: Operation GetCitiesByCountry
 description: "Get all major cities by country name(full / part)."
 operationId: GetCitiesByCountry
 x-ibm-soap:
 soap-action: 'http://www.webserviceX.NET/GetCitiesByCountry'
 soap-operation: '{http://www.webserviceX.NET}GetCitiesByCountry'
 parameters:
 - in: body
 name: body
 required: true
 schema:
 $ref: '#/definitions/GetCitiesByCountryInput'
 responses:
 default:
 description: ''
 schema:
 $ref: '#/definitions/GetCitiesByCountryOutput'
 definitions:
 Security:
 xml:
 namespace: >-
 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
 prefix: wsse
 description: Header for WS-Security
 type: object
 properties:
 UsernameToken:
 xml:
 namespace: >-
 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
 prefix: wsse
 type: object
 properties:
 Username:
 xml:
 namespace: >-
 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
 prefix: wsse
 type: string

```

```

Password:
 xml:
 namespace: >-
 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
 prefix: wsse
 type: string
Nonce:
 xml:
 namespace: >-
 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
 prefix: wsse
 type: string
 properties:
 EncodingType:
 xml:
 namespace: ''
 attribute: true
 type: string
Created:
 xml:
 namespace: >-
 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
 prefix: wsu
 type: string
Timestamp:
 xml:
 namespace: >-
 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
 prefix: wsu
 type: object
 properties:
 Created:
 xml:
 namespace: >-
 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
 prefix: wsu
 type: string
 Expires:
 xml:
 namespace: >-
 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
 prefix: wsu
 type: string
 Id:
 xml:
 namespace: >-
 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
 prefix: wsu
 attribute: true
 type: string
GetWeatherInput:
 description: Input message for wsdl operation GetWeather
 type: object
 properties:
 Envelope:
 xml:
 namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
 prefix: soapenv
 type: object
 properties:
 Header:
 $ref: '#/definitions/GetWeatherHeader'
 Body:
 xml:
 namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
 prefix: soapenv
 type: object
 properties:
 GetWeather:
 $ref: '#/definitions/GetWeather_element_tns'
 required:
 - GetWeather
 required:
 - Body
 required:
 - Envelope
 x-ibm-schema:
 wsdl-port: '{http://www.webserviceX.NET}GlobalWeatherSoap'
 wsdl-operation: GetWeather
 wsdl-message-direction-or-name: GetWeatherRequest
 example: >-

```

```

<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Header>
 <!-- The Security element should be removed if WS-Security is not enabled on the SOAP target-url -->
 <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
 <wsse:UsernameToken>
 <wsse:Username>string</wsse:Username>
 <wsse:Password>string</wsse:Password>
 <wsse:Nonce EncodingType="string">string</wsse:Nonce>
 <wsu:Created>string</wsu:Created>
 </wsse:UsernameToken>
 <wsu:Timestamp wsu:Id="string">
 <wsu:Created>string</wsu:Created>

```

```

 <wsu:Expires>string</wsu:Expires>
 </wsu:Timestamp>
</wsse:Security>
</soapenv:Header>
<soapenv:Body>
 <tns:GetWeather xmlns:tns="http://www.webserviceX.NET"><!-- mandatory -->
 <tns:CityName>string</tns:CityName>
 <tns:CountryName>string</tns:CountryName>
 </tns:GetWeather>
</soapenv:Body>
</soapenv:Envelope>
GetWeatherHeader:
description: Input headers for wsdl operation GetWeather
type: object
properties:
 Security:
 $ref: '#/definitions/Security'
GetWeatherOutput:
description: Output message for wsdl operation GetWeather
type: object
properties:
 Envelope:
 xml:
 namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
 prefix: soapenv
 type: object
 properties:
 Body:
 xml:
 namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
 prefix: soapenv
 type: object
 properties:
 GetWeatherResponse:
 $ref: '#/definitions/GetWeatherResponse_element_tns'
 required:
 - Body
 required:
 - Envelope
x-ibm-schema:
 wsdl-port: '{http://www.webserviceX.NET}GlobalWeatherSoap'
 wsdl-operation: GetWeather
 wsdl-message-direction-or-name: GetWeatherResponse
example: >-

```

```

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Body>
 <tns:GetWeatherResponse xmlns:tns="http://www.webserviceX.NET">
 <tns:GetWeatherResult>string</tns:GetWeatherResult>
 </tns:GetWeatherResponse>
 </soapenv:Body>
</soapenv:Envelope>
GetCitiesByCountryInput:
description: Input message for wsdl operation GetCitiesByCountry
type: object
properties:
 Envelope:
 xml:
 namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
 prefix: soapenv
 type: object
 properties:
 Header:
 $ref: '#/definitions/GetCitiesByCountryHeader'
 Body:
 xml:
 namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
 prefix: soapenv
 type: object
 properties:
 GetCitiesByCountry:
 $ref: '#/definitions/GetCitiesByCountry_element_tns'
 required:
 - GetCitiesByCountry
 required:
 - Body
 required:
 - Envelope
x-ibm-schema:
 wsdl-port: '{http://www.webserviceX.NET}GlobalWeatherSoap'
 wsdl-operation: GetCitiesByCountry
 wsdl-message-direction-or-name: GetCitiesByCountryRequest
example: >-

```

```

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Header>
 <!-- The Security element should be removed if WS-Security is not enabled on the SOAP target-url -->
 <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
 <wsse:UsernameToken>
 <wsse:Username>string</wsse:Username>
 <wsse:Password>string</wsse:Password>
 <wsse:Nonce EncodingType="string">string</wsse:Nonce>
 <wsu:Created>string</wsu:Created>

```

```

 </wsse:UsernameToken>
 <wsu:Timestamp wsu:Id="string">
 <wsu:Created>string</wsu:Created>
 <wsu:Expires>string</wsu:Expires>
 </wsu:Timestamp>
 </wsse:Security>
</soapenv:Header>
<soapenv:Body>
 <tns:GetCitiesByCountry xmlns:tns="http://www.webserviceX.NET"><!-- mandatory -->
 <tns:CountryName>string</tns:CountryName>
 </tns:GetCitiesByCountry>
</soapenv:Body>
</soapenv:Envelope>
GetCitiesByCountryHeader:
 description: Input headers for wsdl operation GetCitiesByCountry
 type: object
 properties:
 Security:
 $ref: '#/definitions/Security'
GetCitiesByCountryOutput:
 description: Output message for wsdl operation GetCitiesByCountry
 type: object
 properties:
 Envelope:
 xml:
 namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
 prefix: soapenv
 type: object
 properties:
 Body:
 xml:
 namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
 prefix: soapenv
 type: object
 properties:
 GetCitiesByCountryResponse:
 $ref: '#/definitions/GetCitiesByCountryResponse_element_tns'
 required:
 - Body
 required:
 - Envelope
 x-ibm-schema:
 wsdl-port: '{http://www.webserviceX.NET}GlobalWeatherSoap'
 wsdl-operation: GetCitiesByCountry
 wsdl-message-direction-or-name: GetCitiesByCountryResponse
 example: >-

```

```

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Body>
 <tns:GetCitiesByCountryResponse xmlns:tns="http://www.webserviceX.NET">
 <tns:GetCitiesByCountryResult>string</tns:GetCitiesByCountryResult>
 </tns:GetCitiesByCountryResponse>
 </soapenv:Body>
</soapenv:Envelope>
GetWeather_element_tns:
 xml:
 namespace: 'http://www.webserviceX.NET'
 prefix: tns
 name: GetWeather
 type: object
 properties:
 CityName:
 xml:
 namespace: 'http://www.webserviceX.NET'
 prefix: tns
 type: string
 CountryName:
 xml:
 namespace: 'http://www.webserviceX.NET'
 prefix: tns
 type: string
 example: |-

```

```

 <tns:GetWeather xmlns:tns="http://www.webserviceX.NET">
 <tns:CityName>string</tns:CityName>
 <tns:CountryName>string</tns:CountryName>
 </tns:GetWeather>
GetWeatherResponse_element_tns:
 xml:
 namespace: 'http://www.webserviceX.NET'
 prefix: tns
 name: GetWeatherResponse
 type: object
 properties:
 GetWeatherResult:
 xml:
 namespace: 'http://www.webserviceX.NET'
 prefix: tns
 type: string
 example: |-

```

```

 <tns:GetWeatherResponse xmlns:tns="http://www.webserviceX.NET">
 <tns:GetWeatherResult>string</tns:GetWeatherResult>
 </tns:GetWeatherResponse>
GetCitiesByCountry_element_tns:

```



```

xml:
 namespace: 'http://www.webserviceX.NET'
 prefix: tns
 name: GetCitiesByCountry
 type: object
 properties:
 CountryName:
 xml:
 namespace: 'http://www.webserviceX.NET'
 prefix: tns
 type: string
 example: |-

```

```

<tns:GetCitiesByCountry xmlns:tns="http://www.webserviceX.NET">
 <tns:CountryName>string</tns:CountryName>
</tns:GetCitiesByCountry>

```

```

GetCitiesByCountryResponse_element_tns:
 xml:
 namespace: 'http://www.webserviceX.NET'
 prefix: tns
 name: GetCitiesByCountryResponse
 type: object
 properties:
 GetCitiesByCountryResult:
 xml:
 namespace: 'http://www.webserviceX.NET'
 prefix: tns
 type: string
 example: |-

```

```

<tns:GetCitiesByCountryResponse xmlns:tns="http://www.webserviceX.NET">
 <tns:GetCitiesByCountryResult>string</tns:GetCitiesByCountryResult>
</tns:GetCitiesByCountryResponse>

```

---

## Example WSDL file: globalweather.wsdl

The following code sample contains a WSDL description.

The WSDL description file globalweather.wsdl is referenced by the example scenario in [Example: Using the platform REST APIs to publish a product containing a SOAP API](#).

```

<wsdl:definitions xmlns:tns="http://www.webserviceX.NET" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
 xmlns:tns="http://www.webserviceX.NET" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
 xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
 targetNamespace="http://www.webserviceX.NET">
 <wsdl:types>
 <s:schema elementFormDefault="qualified" targetNamespace="http://www.webserviceX.NET">
 <s:element name="GetWeather">
 <s:complexType>
 <s:sequence>
 <s:element minOccurs="0" maxOccurs="1" name="CityName" type="s:string" />
 <s:element minOccurs="0" maxOccurs="1" name="CountryName" type="s:string" />
 </s:sequence>
 </s:complexType>
 </s:element>
 <s:element name="GetWeatherResponse">
 <s:complexType>
 <s:sequence>
 <s:element minOccurs="0" maxOccurs="1" name="GetWeatherResult" type="s:string" />
 </s:sequence>
 </s:complexType>
 </s:element>
 <s:element name="GetCitiesByCountry">
 <s:complexType>
 <s:sequence>
 <s:element minOccurs="0" maxOccurs="1" name="CountryName" type="s:string" />
 </s:sequence>
 </s:complexType>
 </s:element>
 <s:element name="GetCitiesByCountryResponse">
 <s:complexType>
 <s:sequence>
 <s:element minOccurs="0" maxOccurs="1" name="GetCitiesByCountryResult" type="s:string" />
 </s:sequence>
 </s:complexType>
 </s:element>
 <s:element name="string" nillable="true" type="s:string" />
 </s:schema>
 </wsdl:types>
 <wsdl:message name="GetWeatherSoapIn">
 <wsdl:part name="parameters" element="tns:GetWeather" />
 </wsdl:message>
 <wsdl:message name="GetWeatherSoapOut">
 <wsdl:part name="parameters" element="tns:GetWeatherResponse" />
 </wsdl:message>

```

```

</wsdl:message>
<wsdl:message name="GetCitiesByCountrySoapIn">
 <wsdl:part name="parameters" element="tns:GetCitiesByCountry" />
</wsdl:message>
<wsdl:message name="GetCitiesByCountrySoapOut">
 <wsdl:part name="parameters" element="tns:GetCitiesByCountryResponse" />
</wsdl:message>
<wsdl:message name="GetWeatherHttpGetIn">
 <wsdl:part name="CityName" type="s:string" />
 <wsdl:part name="CountryName" type="s:string" />
</wsdl:message>
<wsdl:message name="GetWeatherHttpGetOut">
 <wsdl:part name="Body" element="tns:string" />
</wsdl:message>
<wsdl:message name="GetCitiesByCountryHttpGetIn">
 <wsdl:part name="CountryName" type="s:string" />
</wsdl:message>
<wsdl:message name="GetCitiesByCountryHttpGetOut">
 <wsdl:part name="Body" element="tns:string" />
</wsdl:message>
<wsdl:message name="GetWeatherHttpPostIn">
 <wsdl:part name="CityName" type="s:string" />
 <wsdl:part name="CountryName" type="s:string" />
</wsdl:message>
<wsdl:message name="GetWeatherHttpPostOut">
 <wsdl:part name="Body" element="tns:string" />
</wsdl:message>
<wsdl:message name="GetCitiesByCountryHttpPostIn">
 <wsdl:part name="CountryName" type="s:string" />
</wsdl:message>
<wsdl:message name="GetCitiesByCountryHttpPostOut">
 <wsdl:part name="Body" element="tns:string" />
</wsdl:message>
<wsdl:portType name="GlobalWeatherSoap">
 <wsdl:operation name="GetWeather">
 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
 Get weather report for all major cities around the world.
 </wsdl:documentation>
 <wsdl:input message="tns:GetWeatherSoapIn" />
 <wsdl:output message="tns:GetWeatherSoapOut" />
 </wsdl:operation>
 <wsdl:operation name="GetCitiesByCountry">
 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get all major
 cities by country name(full / part).</wsdl:documentation>
 <wsdl:input message="tns:GetCitiesByCountrySoapIn" />
 <wsdl:output message="tns:GetCitiesByCountrySoapOut" />
 </wsdl:operation>
</wsdl:portType>
<wsdl:portType name="GlobalWeatherHttpGet">
 <wsdl:operation name="GetWeather">
 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
 Get weather report for all major cities around the world.
 </wsdl:documentation>
 <wsdl:input message="tns:GetWeatherHttpGetIn" />
 <wsdl:output message="tns:GetWeatherHttpGetOut" />
 </wsdl:operation>
 <wsdl:operation name="GetCitiesByCountry">
 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get all major
 cities by country name(full / part).</wsdl:documentation>
 <wsdl:input message="tns:GetCitiesByCountryHttpGetIn" />
 <wsdl:output message="tns:GetCitiesByCountryHttpGetOut" />
 </wsdl:operation>
</wsdl:portType>
<wsdl:portType name="GlobalWeatherHttpPost">
 <wsdl:operation name="GetWeather">
 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
 Get weather report for all major cities around the world.
 </wsdl:documentation>
 <wsdl:input message="tns:GetWeatherHttpPostIn" />
 <wsdl:output message="tns:GetWeatherHttpPostOut" />
 </wsdl:operation>
 <wsdl:operation name="GetCitiesByCountry">
 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get all major
 cities by country name(full / part).</wsdl:documentation>
 <wsdl:input message="tns:GetCitiesByCountryHttpPostIn" />
 <wsdl:output message="tns:GetCitiesByCountryHttpPostOut" />
 </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="GlobalWeatherSoap" type="tns:GlobalWeatherSoap">
 <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
 <wsdl:operation name="GetWeather">
 <soap:operation soapAction="http://www.webserviceX.NET/GetWeather"
 />
 <wsdl:input>
 <soap:body use="literal" />
 </wsdl:input>
 <wsdl:output>
 <soap:body use="literal" />
 </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="GetCitiesByCountry">
 <soap:operation soapAction="http://www.webserviceX.NET/GetCitiesByCountry"
 />
 <wsdl:input>
 <soap:body use="literal" />
 </wsdl:input>
 </wsdl:operation>

```

```

 <wsdl:output>
 <soap:body use="literal" />
 </wsdl:output>
 </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="GlobalWeatherSoap12" type="tns:GlobalWeatherSoap">
 <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
 <wsdl:operation name="GetWeather">
 <soap12:operation soapAction="http://www.webserviceX.NET/GetWeather"
 />
 <wsdl:input>
 <soap12:body use="literal" />
 </wsdl:input>
 <wsdl:output>
 <soap12:body use="literal" />
 </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="GetCitiesByCountry">
 <soap12:operation soapAction="http://www.webserviceX.NET/GetCitiesByCountry"
 />
 <wsdl:input>
 <soap12:body use="literal" />
 </wsdl:input>
 <wsdl:output>
 <soap12:body use="literal" />
 </wsdl:output>
 </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="GlobalWeatherHttpGet" type="tns:GlobalWeatherHttpGet">
 <http:binding verb="GET" />
 <wsdl:operation name="GetWeather">
 <http:operation location="/GetWeather" />
 <wsdl:input>
 <http:urlEncoded />
 </wsdl:input>
 <wsdl:output>
 <mime:mimeXml part="Body" />
 </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="GetCitiesByCountry">
 <http:operation location="/GetCitiesByCountry" />
 <wsdl:input>
 <http:urlEncoded />
 </wsdl:input>
 <wsdl:output>
 <mime:mimeXml part="Body" />
 </wsdl:output>
 </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="GlobalWeatherHttpPost" type="tns:GlobalWeatherHttpPost">
 <http:binding verb="POST" />
 <wsdl:operation name="GetWeather">
 <http:operation location="/GetWeather" />
 <wsdl:input>
 <mime:content type="application/x-www-form-urlencoded" />
 </wsdl:input>
 <wsdl:output>
 <mime:mimeXml part="Body" />
 </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="GetCitiesByCountry">
 <http:operation location="/GetCitiesByCountry" />
 <wsdl:input>
 <mime:content type="application/x-www-form-urlencoded" />
 </wsdl:input>
 <wsdl:output>
 <mime:mimeXml part="Body" />
 </wsdl:output>
 </wsdl:operation>
</wsdl:binding>
<wsdl:service name="GlobalWeather">
 <wsdl:port name="GlobalWeatherSoap" binding="tns:GlobalWeatherSoap">
 <soap:address location="http://www.webserviceX.com/globalweather.asmx" />
 </wsdl:port>
 <wsdl:port name="GlobalWeatherSoap12" binding="tns:GlobalWeatherSoap12">
 <soap12:address location="http://www.webserviceX.com/globalweather.asmx" />
 </wsdl:port>
 <wsdl:port name="GlobalWeatherHttpGet" binding="tns:GlobalWeatherHttpGet">
 <http:address location="http://www.webserviceX.com/globalweather.asmx" />
 </wsdl:port>
 <wsdl:port name="GlobalWeatherHttpPost" binding="tns:GlobalWeatherHttpPost">
 <http:address location="http://www.webserviceX.com/globalweather.asmx" />
 </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

## API Connect TLS certificates

Details of all Kubernetes TLS certificate objects that are created and used in a default installation of API Connect.

## Overview

TLS is used to secure communications between API Connect pods, subsystems, external services, and users. The TLS certificates that are used by API Connect are created automatically during the installation of API Connect. The certificates are managed by an open source product called [cert-manager](#).

The certificates that are described in this topic all exist as Kubernetes certificate objects in the API Connect namespace where the subsystem that uses the certificate is installed. Run `kubectl get certs -n <namespace>` to see the certificates in your environment. Some of the API Connect Kubernetes certificates are also stored in the management subsystem database. These certificates can be seen in the Cloud Manager UI, in the truststores and keystore of the subsystem default TLS profiles. When the management subsystem `apim` pod is restarted, the certificates in the management database are reloaded from the Kubernetes certificates.

This topic does not cover additional TLS profiles and certificates that API Connect administrators and provider organization owners can create and configure in the Cloud Manager and API Manager UIs. These certificates are stored in the management subsystem database only and do not exist in the Kubernetes environment.

The certificates that are created and used by API Connect are categorized as follows:

- CA certificates - API Connect creates self-signed CA certificates to sign all the API Connect end-entity certificates. The CA certificates are:
  - `ingress-ca` - used by the ingress-issuer. The `ingress-ca` signs all user-facing and inter-subsystem certificates.
  - `management-ca` - signs all the management intra-subsystem certificates.
  - `portal-ca` - signs all the portal intra-subsystem certificates.
  - `analytics-ca` - signs all the analytics intra-subsystem certificates.CA certificates have a default duration of 10 years. All the end-entity certificates have a default duration of 2 years.
- Ingress certificates - API Connect creates a self-signed issuer that is called the ingress-issuer. Ingress certificates are the API Connect certificates that are issued by the ingress-issuer. Ingress certificates are divided into two types:
  - Inter-subsystem certificates - certificates that are used for communication between API Connect subsystems, for example between the management subsystem and gateway. By default, communication between most subsystems uses mTLS. From v10.0.5.3, it is possible to use JWT instead of mTLS: [Using JWT instead of mTLS on Kubernetes and OpenShift](#), [Using JWT instead of mTLS on OVA](#).
  - User-facing certificates - certificates that API Connect users see when they interact with the API Connect UIs and REST API.
- Intra-subsystem certificates - certificates that are used between pods in the same subsystem. Intra-subsystem certificates are signed by a subsystem specific CA certificate: `management-ca`, `portal-ca`, `analytics-ca`. The gateway subsystem is a single pod, and has no intra-subsystem certificates.

On OVA deployments, certificate management is done with the `apicup` command, see [Setting and managing certificates on OVA](#). Do not login to your VMs by `ssh` and attempt to modify certificates directly, unless advised by IBM.

Note: On Cloud Pak for Integration, and OpenShift top-level CR deployments, some certificate names are contracted and prefixed with the `APICConnectCluster` instance name. For example, the certificate `management-ca` is called `<apic instance name>-mgmt-ca`.

## Customizing certificates

It is possible to customize certificates with ones from your own organization. However it is recommended to customize only the user-facing certificates, which are the certificates that your users see when they access the API Connect UIs, CLI, and the REST API.

All other certificates are used internally between API Connect microservices.

Note: The inter and intra-subsystem certificates use of a self-signed root CA presents no security risk. For internal communications with a limited number of predefined endpoints, use of locally generated self-signed CA certificates can be more secure than external CA certificates, as certificates are only ever issued for the known components and the domain of trust is kept as small as possible.

## Renewing certificates

Cert-manager handles the renewal of expired certificates so that the user does not need to monitor or manually renew any of the certificates. However, there are scenarios where the renewal of certificates must be triggered manually. Every API Connect certificate has a corresponding Kubernetes secret object of the same name, deleting this secret triggers cert-manager to re-create the secret with a new TLS certificate. For more information about certificate renewal, see [Renewing TLS certificates](#). Remember: The Kubernetes certificate objects do not contain the TLS certificates themselves, they are pointers to Kubernetes secrets that contain the TLS certificate. See [Key Concepts: Cert-manager, Issuers, and Secrets](#).

For most of the certificates, when they are renewed, the pods that use those certificates are automatically restarted so that they pick up the updated certificate. The minority of certificates that require manual pod restarts after renewal are identified in the reference tables.

If you renew a CA certificate, any end-entity certificates that use this CA certificate must also be renewed.

## Reference tables

The tables show all the API Connect certificates that are created in all possible deployment configurations.

Each table corresponds to a subsystem, and shows the certificates that are used by that subsystem.

Note: On Cloud Pak for Integration or top-level CR deployments, some of the certificates have a shortened name. Where two names for a certificate are shown in the tables, the second name is the name that is used on Cloud Pak for Integration or top-level CR deployments. For example, `management-ca` or `mgmt-ca`.

Table 1. Management certificates

Certificate name	Issuer	Description
------------------	--------	-------------

Certificate name	Issuer	Description
-ingress-ca	selfsigning-issuer	<p>The CA and issuer of all API Connect inter-subsystem and user-facing certificates. If there is a problem with this certificate, then all API Connect subsystems are inaccessible and unable to sync with each other. Update this certificate with kubectl. When updated, all child certificates must also be updated by deleting their corresponding secrets.</p> <p>The ingress-ca certificate is also stored in the management subsystem database. It is visible from the Cloud Manager UI, in the truststores of the analytics, gateway, event gateway, and portal default TLS client profiles. When the management subsystem <code>apim</code> pod is restarted the certificate is reloaded into the database from the Kubernetes ingress-ca certificate. Do not attempt to update this certificate from the Cloud Manager UI.</p> <p>In 2DCDR deployments, and when you have subsystems in different namespaces, it is necessary to manually copy the ingress-ca certificate from the management subsystem to the other subsystems. Steps are provided in the 2DCDR and multi-namespace install sections of this documentation.</p>
management-ca or mgmt-ca	selfsigning-issuer	<p>The issuer for the management subsystems intra-subsystem certificates: management-client, management-server, postgres, and nats certificates. Communication between management subsystem pods fails if there is a problem with this certificate. This certificate is also used as the CA for REST API calls to the management subsystem from the other subsystems, when using <code>in-cluster</code> communication. See <a href="#">In-cluster service communication between subsystems</a></p>
management-client or mgmt-client	management-ca	Client certificate used in communication between management subsystem pods. Communication between management subsystem pods fails if there is a problem with this certificate.
management-server or mgmt-server	management-ca	<p>Server certificate used in communication between management subsystem pods. Communication between management subsystem pods fails if there is a problem with this certificate. Required DNS names in the certificate:</p> <pre>*.&lt;namespace&gt; *.&lt;namespace&gt;.svc *.&lt;instance name&gt;-server.&lt;namespace&gt;.svc &lt;instance name&gt;-server</pre>
analytics-client-client or a7s-cl-client	ingress-issuer	Legacy certificate from pre-v10.0.5 releases. Not used for anything and safe to delete. In v10.0.5, communication with the analytics subsystem uses the <code>analytics-ingestion-client</code> certificate.
analytics-ingestion-client or a7s-ing-client	ingress-issuer	<p>Client certificate used for communication with the analytics subsystem on the ingestion endpoint. This certificate must have:</p> <ul style="list-style-type: none"> <li>The same CA as the server certificate <code>analytics-ai-endpoint</code> on the analytics subsystem.</li> <li>The common name must match the <code>spec.clientSubjectDN</code> in the analytics CR:</li> </ul> <pre>kubectl describe cert analytics-admin-client  Spec:   Common Name:  a7s-ing-client  kubectl get ptl -o yaml  spec:   clientSubjectDN: CN=a7s-ing-client</pre> <p>For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be <code>CN=a7s-ingestion-client</code>, or they could both be <code>CN=a7s-ingestion-client, O=cert-manager</code>, but they must be identical.</p> <p>To update this certificate, use kubectl. Restart the <code>apim</code>, <code>taskmanager</code>, and <code>analytics-proxy</code> pods after the update. This certificate is also used by the gateways for sending API events to the analytics subsystem. The updated certificate is sent by the management subsystem to the gateways, it is not necessary to restart any gateway pods.</p>
portal-admin-client or ptl-adm-client	ingress-issuer	<p>Client certificate used for communication with the portal subsystem on the <code>portalAdminEndpoint</code>. This certificate must have:</p> <ul style="list-style-type: none"> <li>The same CA as the server certificate <code>portal-admin</code> or <code>ptl-director</code> on the portal subsystem.</li> <li>The common name must match the <code>spec.adminClientSubjectDN</code> in the portal CR:</li> </ul> <pre>kubectl describe cert portal-admin-client  Spec:   Common Name:  ptl-adm-client  kubectl get ptl -o yaml  spec:   adminClientSubjectDN: CN=ptl-adm-client</pre> <p>For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be <code>CN=portal-admin-client</code>, or they could both be <code>CN=ptl-adm-client, O=cert-manager</code>, but they must be identical.</p>
gateway-client-client or gw-dr-client	ingress-issuer	Client certificate for communications with the gateway service. Restart the <code>apim</code> , and <code>taskmanager</code> pods after update. This certificate must have the same CA as the <code>gwv6-manager-endpoint</code> or <code>gw-gateway-manager</code> certificate on the gateway.
api-endpoint or mgmt-platform-api	ingress-issuer	Platform API endpoint server certificate. The certificate that is presented to callers of the platform REST API, and to the toolkit CLI.
consumer-endpoint or mgmt-consumer-api	ingress-issuer	Consumer API endpoint server certificate. The certificate presented to callers of the consumer REST API, and to the toolkit CLI.
apim-endpoint or mgmt-api-manager	ingress-issuer	API Manager UI server certificate.

Certificate name	Issuer	Description
cm-endpoint or mgmt-admin	ingress-issuer	Cloud Manager UI server certificate.
db-client-apicuser	management-ca	Intra-subsystem certificate for the management database subsystem.
db-client-pgbouncer	management-ca	Intra-subsystem certificate for the management database subsystem.
db-client-postgres	management-ca	Intra-subsystem certificate for the management database subsystem.
db-client-primaryuser	management-ca	Intra-subsystem certificate for the management database subsystem.
postgres	management-ca	Intra-subsystem certificate for the management database subsystem.
postgres-pgbouncer	management-ca	Intra-subsystem certificate for the management database subsystem.
postgres-operator	management-ca	Intra-subsystem certificate for the management database subsystem.
natscluster-mgmt	management-ca	Intra-subsystem certificate for the nats pods.
db-client-replicator	management-ca	2DCDR deployments only. Client certificate used by the <management_CR>-tunnel pod to connect to the other data center's <management_CR>-tunnel pod.
mgmt-replication-client	ingress-issuer	2DCDR deployments only. Client certificate used in the warm-standby data center's <remote-sitename>-postgres pod to connect to the active data center.
mgmt-replication-server	ingress-issuer	2DCDR deployments only. Server certificate used in the active data center's <management_CR>-tunnel pod. This certificate must contain the DNS Subject Alternative Name of this data center's hostname.
hub-endpoint	ingress-issuer	Used by the Automated API behavior testing application. User-facing server certificate for the hub-endpoint. If there is a problem with this certificate, then the user's browser shows a warning or error message.
turnstile-endpoint	ingress-issuer	Used by the Automated API behavior testing application. User-facing server certificate for the turnstile-endpoint. If there is a problem with this certificate, then the user's browser shows a warning or error message.

Table 2. Analytics certificates

Certificate name	Issuer	Description
analytics-ai-endpoint or a7s-ai-endpoint	ingress-issuer	Server certificate used on the analytics ingestion endpoint, in the mtls-gw pod. Management and gateway subsystems communicate with the analytics subsystem on this endpoint. The client certificates that are used by the management and gateway subsystems must use the same CA certificate as the analytics-ai-endpoint certificate.  If this certificate is updated, restart the mtls-gw pod for the update to take effect.
analytics-ca or a7s-ca	selfsigning-issuer	The issuer for the analytics-client and analytics-server certificates. Communication between analytics subsystem pods fails if there is a problem with this certificate.
analytics-client or a7s-client	analytics-ca	Client certificate used in communication between analytics subsystem pods. Communication between analytics subsystem pods fails if there is a problem with this certificate.
analytics-server or a7s-server	analytics-ca	Server certificate used in communication between analytics subsystem pods. Communication between analytics subsystem pods fails if there is a problem with this certificate. Required DNS names in the certificate:  *.<namespace> *.<namespace>.svc *.<instance name>-server.<namespace>.svc <instance name>-server <instance name>-storage

Table 3. Portal certificates

Certificate name	Issuer	Description
portal-ca or ptl-ca	selfsigning-issuer	The issuer for the portal-client and portal-server certificates. Communication between portal subsystem pods fails if there is a problem with this certificate.
portal-client or ptl-client	portal-ca	Client certificate used in communication between portal subsystem pods. Communication between portal subsystem pods fails if there is a problem with this certificate.
portal-server or ptl-server	portal-ca	Server certificate used in communication between portal subsystem pods. Communication between portal subsystem pods fails if there is a problem with this certificate. Required DNS names in the certificate:  *.<namespace> *.<namespace>.svc *.<instance name>-server.<namespace>.svc <instance name>-server *.<instance name>-<site name>-db-all.<namespace>.svc *.<instance name>-<site name>-www-all.<namespace>.svc *.<instance name>-<site name>-db-all.<namespace>.svc.cluster.local *.<instance name>-<site name>-www-all.<namespace>.svc.cluster.local *.<namespace>.svc.cluster.local <instance name>-db #<remote portal CR name>-db # For 2DCDR only.  <instance name> and <remote portal CR name> are truncated if more than 15 characters.
portal-admin or ptl-director	ingress-issuer	Server certificate used on the portalAdminEndpoint. The management subsystem communicates with the portal subsystem on this endpoint. The corresponding client certificate used in the management to portal communication is the portal-admin-client certificate.  This certificate must have the same CA as the portal-admin-client certificate on the management subsystem.

Certificate name	Issuer	Description
portal-web	ingress-issuer	The server certificate used on the <b>portalUIEndpoint</b> . It is the server certificate that is presented to portal site users, and which their browser authenticates. If there is a problem with this certificate, then users see an error message about an invalid or insecure certificate in their browser when they access a portal site.  The default <b>portal-web</b> certificate is issued by the <b>ingress-issuer</b> , which has a self-signed root certificate. Portal users might see a browser warning about use of a self-signed certificate.
ptl-replication-client	ingress-issuer	2DCDR deployments only. Client certificate that is used in the warm-standby data center's <b>&lt;remote-sitename&gt;-www</b> and <b>&lt;remote-sitename&gt;-db</b> pods to connect to the active data center.
ptl-replication-server	ingress-issuer	2DCDR deployments only. Server certificate used in the active data center's <b>&lt;portal_CR&gt;-tunnel</b> pod, the counterpart to the <b>ptl-replication-client</b> certificate. This certificate must contain the DNS Subject Alternative Name of this data center's hostname.

Table 4. Gateway certificates

Certificate name	Issuer	Description
gateway-peering or gw-peer	ingress-issuer	This certificate secures the communication between gateways in your gateway cluster.
gateway-service or gw-svc	ingress-issuer	Legacy certificate that is not used.
gmv6-endpoint or gw-gateway	ingress-issuer	
gmv6-manager-endpoint or gw-gateway-manager	ingress-issuer	The <b>gatewayManager</b> endpoint certificate. This is the server certificate on the gateway director endpoint, which the management subsystem communicates with. This certificate must be signed by the same CA as the <b>gateway-client-client</b> certificate on management subsystem

Table 5. Event gateway certificates -Cloud Pak for Integration only

Certificate name	Issuer	Description
event-gateway-management-client	ingress-issuer	The <b>event-gateway-management-client</b> certificate exists only on Cloud Pak for Integration deployments.

## TLS certificates organized by usage

Table of API Connect certificates organized by where they are used.

Table 1. Management to gateway communication

Certificate name	Issuer	Description
-ingress-ca	selfsigning-issuer	The CA and issuer of all API Connect inter-subsystem and user-facing certificates. If there is a problem with this certificate, then all API Connect subsystems are inaccessible and unable to sync with each other. Update this certificate with kubectl. When updated, all child certificates must also be updated by deleting their corresponding secrets.  The ingress-ca certificate is also stored in the management subsystem database. It is visible from the Cloud Manager UI, in the truststores of the analytics, gateway, event gateway, and portal default TLS client profiles. When the management subsystem <b>apim</b> pod is restarted the certificate is reloaded into the database from the Kubernetes ingress-ca certificate. Do not attempt to update this certificate from the Cloud Manager UI.  In 2DCDR deployments, and when you have subsystems in different namespaces, it is necessary to manually copy the ingress-ca certificate from the management subsystem to the other subsystems. Steps are provided in the 2DCDR and multi-namespace install sections of this documentation.
gateway-client-client or gw-dr-client	ingress-issuer	Client certificate for communications with the gateway service. Restart the <b>apim</b> , and <b>taskmanager</b> pods after update. This certificate must have the same CA as the <b>gmv6-manager-endpoint</b> or <b>gw-gateway-manager</b> certificate on the gateway.
gmv6-manager-endpoint or gw-gateway-manager	ingress-issuer	The <b>gatewayManager</b> endpoint certificate. This is the server certificate on the gateway director endpoint, which the management subsystem communicates with. This certificate must be signed by the same CA as the <b>gateway-client-client</b> certificate on management subsystem

Table 2. Management to portal communication

Certificate name	Issuer	Description
-ingress-ca	selfsigning-issuer	The CA and issuer of all API Connect inter-subsystem and user-facing certificates. If there is a problem with this certificate, then all API Connect subsystems are inaccessible and unable to sync with each other. Update this certificate with kubectl. When updated, all child certificates must also be updated by deleting their corresponding secrets.  The ingress-ca certificate is also stored in the management subsystem database. It is visible from the Cloud Manager UI, in the truststores of the analytics, gateway, event gateway, and portal default TLS client profiles. When the management subsystem <b>apim</b> pod is restarted the certificate is reloaded into the database from the Kubernetes ingress-ca certificate. Do not attempt to update this certificate from the Cloud Manager UI.  In 2DCDR deployments, and when you have subsystems in different namespaces, it is necessary to manually copy the ingress-ca certificate from the management subsystem to the other subsystems. Steps are provided in the 2DCDR and multi-namespace install sections of this documentation.

Certificate name	Issuer	Description
portal-admin-client or ptl-admin-client	ingress-issuer	<p>Client certificate used for communication with the portal subsystem on the <code>portalAdminEndpoint</code>. This certificate must have:</p> <ul style="list-style-type: none"> <li>The same CA as the server certificate <code>portal-admin</code> or <code>ptl-director</code> on the portal subsystem.</li> <li>The common name must match the <code>spec.adminClientSubjectDN</code> in the portal CR:</li> </ul> <pre>kubectl describe cert portal-admin-client</pre> <pre>Spec:   Common Name:  ptl-admin-client</pre> <pre>kubectl get ptl -o yaml</pre> <pre>spec:   adminClientSubjectDN: CN=ptl-admin-client</pre> <p>For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be <code>CN=portal-admin-client</code>, or they could both be <code>CN=ptl-admin-client</code>, <code>O=cert-manager</code>, but they must be identical.</p>
portal-admin or ptl-director	ingress-issuer	<p>Server certificate used on the <code>portalAdminEndpoint</code>. The management subsystem communicates with the portal subsystem on this endpoint. The corresponding client certificate used in the management to portal communication is the <code>portal-admin-client</code> certificate.</p> <p>This certificate must have the same CA as the <code>portal-admin-client</code> certificate on the management subsystem.</p>

Table 3. Management and gateway to analytics communication

Certificate name	Issuer	Description
-ingress-ca	selfsigning-issuer	<p>The CA and issuer of all API Connect inter-subsystem and user-facing certificates. If there is a problem with this certificate, then all API Connect subsystems are inaccessible and unable to sync with each other. Update this certificate with <code>kubectl</code>. When updated, all child certificates must also be updated by deleting their corresponding secrets.</p> <p>The <code>ingress-ca</code> certificate is also stored in the management subsystem database. It is visible from the Cloud Manager UI, in the truststores of the analytics, gateway, event gateway, and portal default TLS client profiles. When the management subsystem <code>apim</code> pod is restarted the certificate is reloaded into the database from the Kubernetes <code>ingress-ca</code> certificate. Do not attempt to update this certificate from the Cloud Manager UI.</p> <p>In 2DCDR deployments, and when you have subsystems in different namespaces, it is necessary to manually copy the <code>ingress-ca</code> certificate from the management subsystem to the other subsystems. Steps are provided in the 2DCDR and multi-namespace install sections of this documentation.</p>
analytics-ingestion-client or a7s-ing-client	ingress-issuer	<p>Client certificate used for communication with the analytics subsystem on the ingestion endpoint. This certificate must have:</p> <ul style="list-style-type: none"> <li>The same CA as the server certificate <code>analytics-ai-endpoint</code> on the analytics subsystem.</li> <li>The common name must match the <code>spec.clientSubjectDN</code> in the analytics CR:</li> </ul> <pre>kubectl describe cert analytics-admin-client</pre> <pre>Spec:   Common Name:  a7s-ing-client</pre> <pre>kubectl get ptl -o yaml</pre> <pre>spec:   clientSubjectDN: CN=a7s-ing-client</pre> <p>For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be <code>CN=a7s-ingestion-client</code>, or they could both be <code>CN=a7s-ingestion-client</code>, <code>O=cert-manager</code>, but they must be identical.</p> <p>To update this certificate, use <code>kubectl</code>. Restart the <code>apim</code>, <code>taskmanager</code>, and <code>analytics-proxy</code> pods after the update. This certificate is also used by the gateways for sending API events to the analytics subsystem. The updated certificate is sent by the management subsystem to the gateways, it is not necessary to restart any gateway pods.</p>
analytics-ai-endpoint or a7s-ai-endpoint	ingress-issuer	<p>Server certificate used on the analytics ingestion endpoint, in the <code>mtls-gw</code> pod. Management and gateway subsystems communicate with the analytics subsystem on this endpoint. The client certificates that are used by the management and gateway subsystems must use the same CA certificate as the <code>analytics-ai-endpoint</code> certificate.</p> <p>If this certificate is updated, restart the <code>mtls-gw</code> pod for the update to take effect.</p>

Table 4. User-facing certificates

Certificate name	Issuer	Description
api-endpoint or mgmt-platform-api	ingress-issuer	Platform API endpoint server certificate. The certificate that is presented to callers of the platform REST API, and to the toolkit CLI.
consumer-endpoint or mgmt-consumer-api	ingress-issuer	Consumer API endpoint server certificate. The certificate presented that is presented to callers of the consumer REST API, and to the toolkit CLI.
apim-endpoint or mgmt-api-manager	ingress-issuer	API Manager UI server certificate.
cm-endpoint or mgmt-admin	ingress-issuer	Cloud Manager UI server certificate.



Certificate name	Issuer	Description
portal-web	ingress-issuer	The server certificate used on the <code>portalUIEndpoint</code> . It is the server certificate that is presented to portal site users, and which their browser authenticates. If there is a problem with this certificate, then users see an error message about an invalid or insecure certificate in their browser when they access a portal site.  The default <code>portal-web</code> certificate is issued by the <code>ingress-issuer</code> , which has a self-signed root certificate. Portal users might see a browser warning about use of a self-signed certificate.
hub-endpoint	ingress-issuer	Used by the Automated API behavior testing application. User-facing server certificate for the <code>hub-endpoint</code> . If there is a problem with this certificate, then the user's browser shows a warning or error message.
turnstile-endpoint	ingress-issuer	Used by the Automated API behavior testing application. User-facing server certificate for the <code>turnstile-endpoint</code> . If there is a problem with this certificate, then the user's browser shows a warning or error message.

Table 5. Inter-subsystem certificates

Certificate name	Issuer	Description
-ingress-ca	selfsigning-issuer	The CA and issuer of all API Connect inter-subsystem and user-facing certificates. If there is a problem with this certificate, then all API Connect subsystems are inaccessible and unable to sync with each other. Update this certificate with kubectl. When updated, all child certificates must also be updated by deleting their corresponding secrets.  The ingress-ca certificate is also stored in the management subsystem database. It is visible from the Cloud Manager UI, in the truststores of the analytics, gateway, event gateway, and portal default TLS client profiles. When the management subsystem <code>apim</code> pod is restarted the certificate is reloaded into the database from the Kubernetes ingress-ca certificate. Do not attempt to update this certificate from the Cloud Manager UI.  In 2DCDR deployments, and when you have subsystems in different namespaces, it is necessary to manually copy the ingress-ca certificate from the management subsystem to the other subsystems. Steps are provided in the 2DCDR and multi-namespace install sections of this documentation.
analytics-ingestion-client or a7s-ing-client	ingress-issuer	Client certificate used for communication with the analytics subsystem on the ingestion endpoint. This certificate must have: <ul style="list-style-type: none"> <li>The same CA as the server certificate <code>analytics-ai-endpoint</code> on the analytics subsystem.</li> <li>The common name must match the <code>spec.clientSubjectDN</code> in the analytics CR:</li> </ul> <pre>kubectl describe cert analytics-admin-client</pre> <pre>Spec:   Common Name: a7s-ing-client</pre> <pre>kubectl get ptl -o yaml</pre> <pre>spec:   clientSubjectDN: CN=a7s-ing-client</pre> For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be <code>CN=a7s-ingestion-client</code> , or they could both be <code>CN=a7s-ingestion-client, O=cert-manager</code> , but they must be identical.  To update this certificate, use kubectl. Restart the <code>apim</code> , <code>taskmanager</code> , and <code>analytics-proxy</code> pods after the update. This certificate is also used by the gateways for sending API events to the analytics subsystem. The updated certificate is sent by the management subsystem to the gateways, it is not necessary to restart any gateway pods.
portal-admin-client or ptl-adm-client	ingress-issuer	Client certificate used for communication with the portal subsystem on the <code>portalAdminEndpoint</code> . This certificate must have: <ul style="list-style-type: none"> <li>The same CA as the server certificate <code>portal-admin</code> or <code>ptl-director</code> on the portal subsystem.</li> <li>The common name must match the <code>spec.adminClientSubjectDN</code> in the portal CR:</li> </ul> <pre>kubectl describe cert portal-admin-client</pre> <pre>Spec:   Common Name: ptl-adm-client</pre> <pre>kubectl get ptl -o yaml</pre> <pre>spec:   adminClientSubjectDN: CN=ptl-adm-client</pre> For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be <code>CN=portal-admin-client</code> , or they could both be <code>CN=ptl-adm-client, O=cert-manager</code> , but they must be identical.
gateway-client-client or gw-dr-client	ingress-issuer	Client certificate for communications with the gateway service. Restart the <code>apim</code> , and <code>taskmanager</code> pods after update. This certificate must have the same CA as the <code>gwv6-manager-endpoint</code> or <code>gw-gateway-manager</code> certificate on the gateway.
api-endpoint or mgmt-platform-api	ingress-issuer	Platform API endpoint server certificate. The certificate that is presented to callers of the platform REST API, and to the toolkit CLI.
consumer-endpoint or mgmt-consumer-api	ingress-issuer	Consumer API endpoint server certificate. The certificate presented that is presented to callers of the consumer REST API, and to the toolkit CLI.
apim-endpoint or mgmt-api-manager	ingress-issuer	API Manager UI server certificate.
cm-endpoint or mgmt-admin	ingress-issuer	Cloud Manager UI server certificate.

Certificate name	Issuer	Description
mgmt-replication-client	ingress-issuer	2DCDR deployments only. Client certificate used in the warm-standby data center's <remote-sitename>-postgres pod to connect to the active data center.
mgmt-replication-server	ingress-issuer	2DCDR deployments only. Server certificate used in the active data center's <management_CR>-tunnel pod. This certificate must contain the DNS Subject Alternative Name of this data center's hostname.
hub-endpoint	ingress-issuer	Used by the Automated API behavior testing application. User-facing server certificate for the hub-endpoint. If there is a problem with this certificate, then the user's browser shows a warning or error message.
turnstile-endpoint	ingress-issuer	Used by the Automated API behavior testing application. User-facing server certificate for the turnstile-endpoint. If there is a problem with this certificate, then the user's browser shows a warning or error message.
analytics-ai-endpoint or a7s-ai-endpoint	ingress-issuer	Server certificate used on the analytics ingestion endpoint, in the mtls-gw pod. Management and gateway subsystems communicate with the analytics subsystem on this endpoint. The client certificates that are used by the management and gateway subsystems must use the same CA certificate as the analytics-ai-endpoint certificate.  If this certificate is updated, restart the mtls-gw pod for the update to take effect.
portal-admin or ptl-director	ingress-issuer	Server certificate used on the portalAdminEndpoint. The management subsystem communicates with the portal subsystem on this endpoint. The corresponding client certificate used in the management to portal communication is the portal-admin-client certificate.  This certificate must have the same CA as the portal-admin-client certificate on the management subsystem.
portal-web	ingress-issuer	The server certificate used on the portalUIEndpoint. It is the server certificate that is presented to portal site users, and which their browser authenticates. If there is a problem with this certificate, then users see an error message about an invalid or insecure certificate in their browser when they access a portal site.  The default portal-web certificate is issued by the ingress-issuer, which has a self-signed root certificate. Portal users might see a browser warning about use of a self-signed certificate.
ptl-replication-client	ingress-issuer	2DCDR deployments only. Client certificate that is used in the warm-standby data center's <remote-sitename>-www and <remote-sitename>-db pods to connect to the active data center.
ptl-replication-server	ingress-issuer	2DCDR deployments only. Server certificate used in the active data center's <portal_CR>-tunnel pod, the counterpart to the ptl-replication-client certificate. This certificate must contain the DNS Subject Alternative Name of this data center's hostname.
gateway-peering or gw-peer	ingress-issuer	This certificate secures the communication between gateways in your gateway cluster.
gwv6-endpoint or gw-gateway	ingress-issuer	
gwv6-manager-endpoint or gw-gateway-manager	ingress-issuer	The gatewayManager endpoint certificate. This is the server certificate on the gateway director endpoint, which the management subsystem communicates with. This certificate must be signed by the same CA as the gateway-client-client certificate on management subsystem
event-gateway-management-client	ingress-issuer	The event-gateway-management-client certificate exists only on Cloud Pak for Integration deployments.

Table 6. Intra-subsystem certificates

Certificate name	Issuer	Description
management-ca or mgmt-ca	selfsigning-issuer	The issuer for the management subsystems intra-subsystem certificates: management-client, management-server, postgres, and nats certificates. Communication between management subsystem pods fails if there is a problem with this certificate. This certificate is also used as the CA for REST API calls to the management subsystem from the other subsystems, when using in-cluster communication. See <a href="#">In-cluster service communication between subsystems</a>
management-client or mgmt-client	management-ca	Client certificate used in communication between management subsystem pods. Communication between management subsystem pods fails if there is a problem with this certificate.
management-server or mgmt-server	management-ca	Server certificate used in communication between management subsystem pods. Communication between management subsystem pods fails if there is a problem with this certificate. Required DNS names in the certificate: * .<namespace> * .<namespace>.svc * .<instance name>-server.<namespace>.svc <instance name>-server
db-client-apicuser	management-ca	Intra-subsystem certificate for the management database subsystem.
db-client-pgbouncer	management-ca	Intra-subsystem certificate for the management database subsystem.
db-client-postgres	management-ca	Intra-subsystem certificate for the management database subsystem.
db-client-primaryuser	management-ca	Intra-subsystem certificate for the management database subsystem.
postgres	management-ca	Intra-subsystem certificate for the management database subsystem.
postgres-pgbouncer	management-ca	Intra-subsystem certificate for the management database subsystem.
postgres-operator	management-ca	Intra-subsystem certificate for the management database subsystem.
natscluster-mgmt	management-ca	Intra-subsystem certificate for the nats pods.
db-client-replicator	management-ca	2DCDR deployments only. Client certificate used by the <management_CR>-tunnel pod to connect to the other data center's <management_CR>-tunnel pod.
analytics-ca or a7s-ca	selfsigning-issuer	The issuer for the analytics-client and analytics-server certificates. Communication between analytics subsystem pods fails if there is a problem with this certificate.

Certificate name	Issuer	Description
<code>analytics-client</code> or <code>a7s-client</code>	<code>analytics-ca</code>	Client certificate used in communication between analytics subsystem pods. Communication between analytics subsystem pods fails if there is a problem with this certificate.
<code>analytics-server</code> or <code>a7s-server</code>	<code>analytics-ca</code>	Server certificate used in communication between analytics subsystem pods. Communication between analytics subsystem pods fails if there is a problem with this certificate. Required DNS names in the certificate: <pre>*.&lt;namespace&gt; *.&lt;namespace&gt;.svc *.&lt;instance name&gt;-server.&lt;namespace&gt;.svc &lt;instance name&gt;-server &lt;instance name&gt;-storage</pre>
<code>portal-ca</code> or <code>ptl-ca</code>	<code>selfsigning-issuer</code>	The issuer for the portal-client and portal-server certificates. Communication between portal subsystem pods fails if there is a problem with this certificate.
<code>portal-client</code> or <code>ptl-client</code>	<code>portal-ca</code>	Client certificate used in communication between portal subsystem pods. Communication between portal subsystem pods fails if there is a problem with this certificate.
<code>portal-server</code> or <code>ptl-server</code>	<code>portal-ca</code>	Server certificate used in communication between portal subsystem pods. Communication between portal subsystem pods fails if there is a problem with this certificate. Required DNS names in the certificate: <pre>*.&lt;namespace&gt; *.&lt;namespace&gt;.svc *.&lt;instance name&gt;-server.&lt;namespace&gt;.svc &lt;instance name&gt;-server *.&lt;instance name&gt;-&lt;site name&gt;-db-all.&lt;namespace&gt;.svc *.&lt;instance name&gt;-&lt;site name&gt;-www-all.&lt;namespace&gt;.svc *.&lt;instance name&gt;-&lt;site name&gt;-db-all.&lt;namespace&gt;.svc.cluster.local *.&lt;instance name&gt;-&lt;site name&gt;-www-all.&lt;namespace&gt;.svc.cluster.local *.&lt;namespace&gt;.svc.cluster.local &lt;instance name&gt;-db #&lt;remote portal CR name&gt;-db # For 2DCDR only.</pre> <i>&lt;instance name&gt; and &lt;remote portal CR name&gt; are truncated if more than 15 characters.</i>

## Troubleshooting API Connect

This section provides some general troubleshooting steps for use when encountering common problems with API Connect.

### About

The following sections provide advice on diagnosing some common problems. This information can help you to identify whether a specific API Connect component is failing, whether it's an environmental problem, and whether you should raise a support request. Some advice includes reviewing specific log files so you can narrow down the source of the problem.

- [Unable to access Management UIs](#)
- [Unable to access the Portal sites](#)
- [Login failure on Management UIs](#)
- [Login failure on Portal sites](#)
- [Management operations not reflected on Gateway or Portal](#)

### Related tasks

- [Gathering post-mortem logs](#)
- [Gathering logs for a VMware environment](#)

### Related information

- [IBM Support Service Requests and PMRs](#)

## Unable to access Management UIs

If you are unable to access the UI login page, getting a blank page, an HTTP 404, or some other error.

When this happens it means that either the Management subsystem is down, or the access attempt is not reaching the Management subsystem due to a problem in the environment.

Note: For OVA users some of the below checks involve running kubectl commands on your VMs. To access your VMs for running kubectl commands:

1. ssh to one of your Management VMs as the user 'apicadm', for example: `ssh apicadm@mgmt1.example.com`
2. Sudo to the root user: `sudo -i`
3. From here you will be able to run kubectl commands, for example:

```
kubectl get pods
NAME READY STATUS RESTARTS AGE
```

```
abc-management-analytics-proxy-5f8bcd74c4-jbhch 1/1 Running 1 (97d ago) 113d
...
```

4. API Connect pods run in the default namespace on OVA, so there is no need to specify the namespace with `-n` in the kubectl commands.

Check the following:

- Is the access attempt actually reaching the Management subsystem? To confirm this, check what is in the logs of the `juhu` pods which run on the Management subsystem. The `juhu` pods are the entry point for all UI, REST, and CLI requests to the Management subsystem. For every access request there should be line logged by the `juhu` pod, for example: When the user attempts to access the admin UI login page, the following is logged by the `juhu` pod:

```
kubectl logs abc-management-juhu-fc8f6bb5d-jnw2q --since=5m
...
9.20.85.205 - - [27/May/2022:10:35:49 +0000] [675fa59030a1a68028712ad2fdb2fdb1] "GET /api/cloud/admin/identity-providers
HTTP/1.1" 200 301 "https://apimsvc20.hursley.ibm.com/auth/admin/sign-in/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safari/537.36" "9.145.13.216" [127.0.0.1:2000]
request_time=0.222 upstream_response_time=0.224
...
```

Note that the timestamp of these log records are not the time when the request arrived, but the time when the response to the request was sent back. To work out the arrival time you can subtract the `request_time=0.222` that appears later in the log message. If there is nothing logged in the `juhu` pods at the time the UI access is attempted, then check if you have any network problems between your browser and the Management subsystem. If you are on OVA then the `tcpdump` utility is available which you can use to confirm if traffic is reaching your Management VM, for example `tcpdump -i any src host 9.20.86.22 and dst port 443`

- Are the Management subsystem pods all up and running? Check this with `kubectl get ManagementCluster -n <namespace>`. If this does not show all pods running, then check which pods are failing: `kubectl get pods -n <namespace>`. If you are on OVA then gather postmortem logs and open a support case at this point. Otherwise it's worth checking the reasons why the pods are failing to start with `kubectl describe` as it might be something in your kubernetes environment that you can fix. Note that on the Management subsystem the `postgres` pods will need to be running first, so if these are not running then most other Management pods will not start.
- Check the logs of the `apim` pods while attempting to access the UI. The `juhu` pods receive the requests from the UI, and send them on to the `apim` pods for processing. The logs of the `apim` pods may reveal some clues as to the cause of the problem. If nothing is logged in the `apim` pods at the time of the access attempt then it suggests a problem in the environment, the `juhu` pods unable to forward requests to `apim`.

## Related tasks

- [Gathering post-mortem logs](#)
- [Gathering logs for a VMware environment](#)

## Related information

- [IBM Support Service Requests and PMRs](#)

## Unable to access the Portal sites

If you are unable to access the login page of a Portal site, getting a blank page, an HTTP 404, or some other error.

When this happens it means that either the Portal site is down, the Portal subsystem is down, or the access attempt is not reaching the Portal due to a problem in the environment.

Note: For OVA users some of the below checks involve running kubectl commands on your VMs. To access your VMs for running kubectl commands:

1. ssh to one of your Management VMs as the user 'apicadm', for example: `ssh apicadm@mgmt1.example.com`
2. Sudo to the root user: `sudo -i`
3. From here you will be able to run kubectl commands, for example:

```
kubectl get pods
NAME READY STATUS RESTARTS AGE
abc-management-analytics-proxy-5f8bcd74c4-jbhch 1/1 Running 1 (97d ago) 113d
...
```

4. API Connect pods run in the default namespace on OVA, so there is no need to specify the namespace with `-n` in the kubectl commands.

Check the following:

- Is the access attempt actually reaching the Portal subsystem? To confirm this, check what is in the logs of the `web` container which runs in the `www` pods which run on the Portal subsystem. The `web` containers are where the interactions between users and the Portal sites are logged. Specifically the `nginx` process that runs in the `web` container is the entry point for site access. For every attempt to access a portal site there should be line logged by the `web` container, for example: When the user attempts to access a sites home page, the following is logged by the `web` container:

```
kubectl logs abc-portal-sitel-www-0 -c web | grep nginx
...
[nginx stdout] 584 ce6ce0:adad58:00580c 2022-05-27 12:57:38: "9.145.11.161, 9.20.85.206" portal.site1.example.com
"GET /providerorg1/sandbox/ HTTP/1.1" 200 5026 5315 5992 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safari/537.36" 0.330 "-"
...
```

If there is nothing logged by the `web` containers `nginx` process at the time the site access is attempted, then check if you have any network problems between your browser and the Portal subsystem, or problems with kubernetes ingresses/OCP routes to the portal web endpoint. If you are on OVA then the `tcpdump` utility is available which you can use to confirm if traffic is reaching your Portal VM, for example `tcpdump -i any src host 9.20.86.22 and dst port 443`

- Are the Portal subsystem pods all up and running? Check this with `kubectl get PortalCluster -n <namespace>`. If this does not show all pods running, then check which pods are failing: `kubectl get pods -n <namespace>`. If you are on OVA then gather postmortem logs and open a support case at this point. Otherwise it's worth checking the reasons why the pods are failing to start with `kubectl describe` as it might be something in your kubernetes environment that you can fix. Note that on the Portal subsystem the `-db-` pods will need to be running first, so if these are not running then the other Portal pods will not start.
- If the `web` container logs show the access attempt being received, there may be clues as to the problem in what is logged immediately after this.

## Related tasks

- [Gathering post-mortem logs](#)
- [Gathering logs for a VMware environment](#)

## Related information

- [IBM Support Service Requests and PMRs](#)

## Login failure on Management UIs

If you are able to access the Management UIs login page, but the login attempt is failing.

Note: For OVA users some of the below checks involve running kubectl commands on your VMs. To access your VMs for running kubectl commands:

1. ssh to one of your Management VMs as the user 'apicadm', for example: `ssh apicadm@mgmt1.example.com`
2. Sudo to the root user: `sudo -i`
3. From here you will be able to run kubectl commands, for example:

```
kubectl get pods
NAME READY STATUS RESTARTS AGE
abc-management-analytics-proxy-5f8bcd74c4-jbhch 1/1 Running 1 (97d ago) 113d
...
```

4. API Connect pods run in the default namespace on OVA, so there is no need to specify the namespace with `-n` in the kubectl commands.

Check the logs of the `juhu` and `apim` pods at the time of the login attempt to confirm that the login attempt did reach the Management subsystem. For more information on the `juhu` logs see [Unable to access Management UIs](#). If the login attempt reached the Management subsystem, more details of the login failure for Local User Registry users can be found in the `lur` pod logs, and for LDAP users in the `ldap` pods. Example:

```
kubectl logs abc-management-lur-8bbc47b54-9v18q --since=1m -f
...
2022-05-27T13:41:13.511Z audit [4d55be1800dcef18c64e0470484e6eac] START: >>>>>>> [POST]: /user-login
2022-05-27T13:41:13.704Z lur:error [4d55be1800dcef18c64e0470484e6eac] Requested login, but user, notouser not found in user
registry,
2022-05-27T13:41:13.706Z bhendi:error [4d55be1800dcef18c64e0470484e6eac] Error in POST post:/user-login (user.js:login)
- status : 401
...
```

## Related tasks

- [Gathering post-mortem logs](#)
- [Gathering logs for a VMware environment](#)

## Related information

- [IBM Support Service Requests and PMRs](#)

## Login failure on Portal sites

If you are able to access the Portal sites login page, but the login attempt is failing.

Note: For OVA users some of the below checks involve running kubectl commands on your VMs. To access your VMs for running kubectl commands:

1. ssh to one of your Management VMs as the user 'apicadm', for example: `ssh apicadm@mgmt1.example.com`
2. Sudo to the root user: `sudo -i`
3. From here you will be able to run kubectl commands, for example:

```
kubectl get pods
NAME READY STATUS RESTARTS AGE
abc-management-analytics-proxy-5f8bcd74c4-jbhch 1/1 Running 1 (97d ago) 113d
...
```

4. API Connect pods run in the default namespace on OVA, so there is no need to specify the namespace with `-n` in the kubectl commands.

All Portal users except for the site Admin user are ultimately authenticated by the Management subsystem, however Portal adds additional security controls such as IP banning and Flood control which should be first place to check for portal login issues: [Managing Developer Portal security](#).

If the login problem is with the Admin user, then check the logs of the `web` container in the `www` pod at the time of the attempt, for example:

```
kubectl logs abc-portal-site1-www-0 -c web --since=1m
...
[php-fpm stderr] 585 ce6ce0:adad58:c3852e 2022-05-27 14:16:49: WARNING: [pool www] child 1023 said into stderr: "WATCHDOG:
[NOTICE] [user] Login attempt failed for admin. | user: anonymous | uri:
https://provorg1.sandbox.portal.apimsvc21.hursley.ibm.com/provorg1/sandbox/user/login | referer:
https://portal.site1.example.com/provorg1/sandbox/user/login
...
```

If the Portal site itself is not rejecting the user login, then check the Management server logs during the login attempt: [Login failure on Management UIs](#). It may also be worth checking the logs of the `admin` container in the `www` pod, which manages the Portal end of the communications with the Management subsystem.

Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address `alice@acme.com` can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: **A user already exists with this email address.**

---

## Management operations not reflected on Gateway or Portal

Where publishes or deletes executed on the Management subsystem are not reflected on the gateway or portal within a few minutes. These problems are often caused by a network failure between Management and the other subsystems, or due to TLS handshaking failures.

Note: For OVA users some of the below checks involve running `kubectl` commands on your VMs. To access your VMs for running `kubectl` commands:

1. ssh to one of your Management VMs as the user 'apicadm', for example: `ssh apicadm@mgmt1.example.com`
2. Sudo to the root user: `sudo -i`
3. From here you will be able to run `kubectl` commands, for example:

```
kubectl get pods
NAME READY STATUS RESTARTS AGE
abc-management-analytics-proxy-5f8bcd74c4-jbhch 1/1 Running 1 (97d ago) 113d
...
```

4. API Connect pods run in the default namespace on OVA, so there is no need to specify the namespace with `-n` in the `kubectl` commands.

For Gateway, the first place to check is the processing status page: [Reviewing a gateway's processing status](#)

For Portal and Gateway check the logs of the `taskmanager` pods at the time of the publish attempt. These pods send the publish operations to the gateway and portals via webhooks. If there is a network problem you may see socket timeout errors in the logs of these pods, or TLS errors if there is problem with your certificates. A common problem are load-balancers doing TLS termination instead of passthrough (where JWT security is not used, see [Using JWT instead of mTLS on Kubernetes and OpenShift](#), [Using JWT instead of mTLS on OVA](#)). If the logs suggest that the webhook was sent successfully then check the gateway director (`gwd`) logs on the Gateway, and the `www` pod `admin` container logs on the portal, for example:

```
kubectl logs abc-portal-site1-www-0 -c admin
```

Taskmanager pods execute 'send' tasks to publish operations to the gateways and portals. The taskmanager pods log messages about the progress of tasks, so there should be messages in a taskmanager pod for a send task soon after applying a change on the Management subsystem. A "send running in pod" message like the following example means the send task is running:

```
2023-05-05 14:02:37.270 management-taskmanager-f5d6fd77-dmnn taskmanager apim:taskmanager:info:taskProcessor [0d30189c-db39-4260-bb1f-a75a3ac5f1ba] task id: b3933261-5799-473d-9b83-9e29c245fde8 / kind: send running in pod: management-taskmanager-f5d6fd77-dmnn / containerId: cri-o://d5afe6d3960e333c1ea5d8f95b79564e769bc2ed5de9d8baebdb84b3f3ad4be 269ms after claimed
```

If the taskmanager logs an error containing 'dispatching task failed for task' (logged when attempting to start a task) or an error containing 'Stale claimed task' (logged 15 minutes later), API Connect might have a problem running tasks. A small number of these errors might be reported from time to time, and the system recovers automatically. But if the errors continue being reported, restart the `natscluster` pods (all together) and then restart the `taskmanager` pods to recover by running the following commands:

```
kubectl -n <namespace> delete po -l app.kubernetes.io/name=natscluster
kubectl -n <namespace> delete po -l app.kubernetes.io/name=taskmanager
```

When raising a support case be sure to include logs from both the Management subsystem and the affected subsystems gateway/portal, stating when the publish attempt was made.

---

## Related information

- [IBM Support Service Requests and PMRs](#)

---

## Advanced maintenance operations

Topics covering advanced operations that are applicable to all supported platforms.

---

## Management subsystem Postgres failover steps

If your Postgres leader pod is not the original Postgres leader, complete the failover steps to restore the original pod as leader.

## Before you begin

This topic applies only to three replica deployments of API Connect.

Follow the steps in this topic if one or more of the following conditions are reported:

- If the **apicops version:pre-upgrade** command reports:

```
Current postgres leader is not the original leader
```

- Openshift and Cloud Pak for Integration only: Your API Connect cluster CR reports **Error** status, and the output of:

```
oc get apiconnectcluster -n <namespace> -o yaml
```

shows:

```
Original PostgreSQL primary is running as replica, please perform failover to original primary before attempting upgrade.
Refer - https://ibm.biz/BdPLWU
```

- When you attempt to upgrade the management subsystem by editing the management CR to change the version and license and you see this error:

```
Original PostgreSQL primary is running as replica, please perform failover to original primary before attempting upgrade.
Refer - https://ibm.biz/BdPLWU
```

## About this task

The management subsystem uses Postgres as the database. A three replica API Connect deployment has a Postgres cluster that consists of 1 Postgres leader, and 2 Postgres replicas.

Note: In Kubernetes terminology, every pod in a cluster is called a "replica". In Postgres terminology, a Postgres cluster consists of "members", one of the members is designated as the leader, and the others are designated as "replicas". During normal operation, the Postgres cluster member that is the leader can change; one of the Postgres replicas can become the leader, and the original leader become a replica.

If the current Postgres cluster leader in your management deployment is not the original leader, then problems can occur during upgrade. To prevent upgrade problems, follow the steps in this topic to identify the original Postgres leader and assign it as the leader again.

Note: If you have an OVA deployment, run the steps that are documented in this topic from your management virtual appliance, as the root user: Login to your management virtual appliance with an SSH client and the username **apicadm**, then switch to the root user with **sudo**

-i.

## Procedure

- Assign your management subsystem namespace to a new environment variable called **\$NAMESPACE**

```
export NAMESPACE=<management namespace>
```

Note: On OVA deployments, set this to "default":

```
export NAMESPACE=default
```

- Identify the original name of your Postgres cluster, and assign it to a new environment variable **\$PGCLUSTER\_NAME**.

```
export PGCLUSTER_NAME=$(kubectl -n $NAMESPACE get pgcluster | awk 'NR==2{print $1}')
```

Confirm the variable is set correctly with:

```
echo $PGCLUSTER_NAME
```

- Get a list of the PVCs whose names start with **\$PGCLUSTER\_NAME** in your API Connect namespace.

```
kubectl -n $NAMESPACE get pvc $PGCLUSTER_NAME $PGCLUSTER_NAME-wal
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
m1-33253de3-postgres	Bound	local-pv-2b61c0bd	234Gi	RWO	local-storage	6h42m
m1-33253de3-postgres-wal	Bound	local-pv-bd893695	234Gi	RWO	local-storage	6h42m

Note: If no PVCs are returned then check the **apicops** error or management CR warning, if it says "The original primary PVC is not attached to any pods in this namespace", then do not proceed with these steps. Instead, to re-create your Postgres cluster with a new leader:

- Take a new management database backup [Backup and restore](#).
- Restore the management database from the new backup.

- Get the names of all the Postgres pods.

```
kubectl get pods -n $NAMESPACE -l 'role in (master,replica)'
```

NAME	READY	STATUS	RESTARTS	AGE
m1-33253de3-postgres-dbd7966db-1tfhq	1/1	Running	0	61s
m1-33253de3-postgres-dxxx-c54776d87-wldgb	1/1	Running	0	26m
m1-33253de3-postgres-xwjf-8577b6c669-9chcp	1/1	Running	0	26m

- Identify which pod was the original Postgres leader.

Describe each Postgres pod to see which one is using the two PVCs identified in step 3.

```
kubectl -n $NAMESPACE describe pod <pod-name> | grep 'ClaimName'
```

```
k describe pod m1-33253de3-postgres-dbd7966db-1tfhq | grep ClaimName
ClaimName: m1-33253de3-postgres
ClaimName: m1-33253de3-postgres-wal
```

```
k describe pod m1-33253de3-postgres-dxxx-c54776d87-wldgb | grep ClaimName
```

```
ClaimName: m1-33253de3-postgres-dxxx
ClaimName: m1-33253de3-postgres-dxxx-wal
```

```
k describe pod m1-33253de3-postgres-xwjf-8577b6c669-9chcp | grep ClaimName
ClaimName: m1-33253de3-postgres-xwjf
ClaimName: m1-33253de3-postgres-xwjf-wal
```

In this example, the highlighted pod `m1-33253de3-postgres-dbd7966db-1tfhq` is the pod that is using the PVCs identified in step 3. Set this pod to the environment variable `$ORIGINAL_POSTGRES_LEADER`:

```
export ORIGINAL_POSTGRES_LEADER=<pod-name>
```

Note: If none of the Postgres pods are using the PVCs identified in step 3, then do not proceed with these steps. Instead, to re-create your Postgres cluster with a new leader:

- a. Take a new management database backup [Backup and restore](#).
- b. Restore the management database from the new backup.

6. Run the `patronictl list` command in each Postgres pod and examine the output.

```
kubectl -n $NAMESPACE exec -it <postgres-pod> -- patronictl list
```

Example output for each Postgres pod:

```
Defaulted container "database" out of: database, set-libpq-certs (init)
+-----+-----+-----+-----+-----+-----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| Cluster: m1-33253de3-postgres (7039951359768572098) |
+-----+-----+-----+-----+-----+-----+
| m1-33253de3-postgres-dbd7966db-1tfhq | 192.168.51.112 | Replica | running | 8 | 0 |
| m1-33253de3-postgres-dxxx-c54776d87-wldgb | 192.168.112.17 | Replica | running | 8 | 0 |
| m1-33253de3-postgres-xwjf-8577b6c669-9chcp | 192.168.117.219 | Leader | running | 8 | 0 |
+-----+-----+-----+-----+-----+-----+
```

Each Postgres pod should show the same output, although the order of the records might differ. The names in the `Member` column should match the names of the Postgres pods. Check the following:

- If the original Postgres pod that you identified in step 5 has a `Lag in MB` of zero and a `TL` the same as the `Leader` pod, then continue to step 7.
- If the pod labeled `Leader` shows a `TL` that is less than the other pods, or a non-null `Lag in MB`, then proceed no further and open a support case.
- If the original Postgres pod that you identified in step 5 shows a lower `TL` than the other pods, or a nonzero value in `Lag in MB`, then reinitialize that pod:

```
kubectl exec -it $ORIGINAL_POSTGRES_LEADER -n $NAMESPACE -- patronictl reinit <cluster name>
$ORIGINAL_POSTGRES_LEADER
```

where:

- `<cluster name>` is the name of the Postgres cluster that is shown in the `patronictl list` output:

```
+ Cluster: <cluster name> ...
```

Run `patronictl list` again to confirm that the `Lag in MB` is reducing. Do not proceed until it is zero. If it is not reducing, open a support case.

7. Make the pod that you identified in step 5 the `Leader`.

Run:

```
kubectl -n $NAMESPACE exec -it <postgres pod> -- patronictl failover
```

and select your `$ORIGINAL_POSTGRES_LEADER` as the candidate to failover to:

```
Candidate ['m1-33253de3-postgres-dbd7966db-1tfhq', 'm1-33253de3-postgres-dxxx-c54776d87-wldgb'] []: m1-33253de3-postgres-dbd7966db-1tfhq
Current cluster topology
+-----+-----+-----+-----+-----+-----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| Cluster: m1-33253de3-postgres (7039951359768572098) |
+-----+-----+-----+-----+-----+-----+
| m1-33253de3-postgres-dbd7966db-1tfhq | 192.168.51.112 | Replica | running | 8 | 0 |
| m1-33253de3-postgres-dxxx-c54776d87-wldgb | 192.168.112.17 | Replica | running | 8 | 0 |
| m1-33253de3-postgres-xwjf-8577b6c669-9chcp | 192.168.117.219 | Leader | running | 8 | 0 |
+-----+-----+-----+-----+-----+-----+
Are you sure you want to failover cluster m1-33253de3-postgres, demoting current master m1-33253de3-postgres-xwjf-8577b6c669-9chcp? [y/N]: y
2021-12-10 22:07:02.15154 Successfully failed over to "m1-33253de3-postgres-dbd7966db-1tfhq"
+-----+-----+-----+-----+-----+-----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| Cluster: m1-33253de3-postgres (7039951359768572098) |
+-----+-----+-----+-----+-----+-----+
| m1-33253de3-postgres-dbd7966db-1tfhq | 192.168.51.112 | Leader | running | 8 | 0 |
| m1-33253de3-postgres-dxxx-c54776d87-wldgb | 192.168.112.17 | Replica | running | 8 | 0 |
| m1-33253de3-postgres-xwjf-8577b6c669-9chcp | 192.168.117.219 | Replica | stopped | 8 | unknown |
+-----+-----+-----+-----+-----+-----+
```

8. Run `patronictl list` to confirm that the pod that was identified in step 5 is the new leader, and that `TL` and `Lag in MB` are reported to be the same in each pod.

```
kubectl exec -it <postgres-pod> -n $NAMESPACE -- patronictl list
```

```
+-----+-----+-----+-----+-----+-----+
| Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| Cluster: m1-33253de3-postgres (7039951359768572098) |
+-----+-----+-----+-----+-----+-----+
| m1-33253de3-postgres-dbd7966db-1tfhq | 192.168.51.112 | Leader | running | 9 | 0 |
| m1-33253de3-postgres-dxxx-c54776d87-wldgb | 192.168.112.17 | Replica | running | 9 | 0 |
| m1-33253de3-postgres-xwjf-8577b6c669-9chcp | 192.168.117.219 | Replica | running | 9 | 0 |
+-----+-----+-----+-----+-----+-----+
```